차 례

머 리 말

위대한 령도자 **김정일**원수님께서는 다음과 같이 지적하시였다.

《프로그람을 개발하는데서 기본은 우리 식의 프로그람을 개발하는것입니다. 우리는 우리 식의 프로그람을 개발하는 방향으로 나가야 합니다.》

프로그람기술을 빨리 발전시키는것은 나라의 정보기술을 높은 수준에 올려세우기 위한 중요한 요구의 하나이다.

위대한 령도자 **김정일**원수님의 현명한 령도에 의하여 오늘 우리 나라에는 정보기술을 빨리 발전시킬수 있는 물질기술적토대가 튼튼히 마련되였으며 새로운 정보기술성과들이 련이어 이룩되고있다.

나라의 정보기술을 높은 수준에 올려세우는데서 중요한것은 우리 식 조작체계인 《붉은별》에서 응용할수 있는 프로그람들을 빨리 발전시키는것이다.

6학년 《콤퓨터》에서는 아래학년에서 배운 간단한 프로그람작성지식에 기초하여 자료관리 및 함수에 의한 프로그람작성과 도형그리기와 같은 여러가지 프로그람작성방법들에 대하여 배우게 된다.

먼저 배렬과 문자렬, 구조체와 지적자 등과 같은 구조적인 변수들에 대한 개념과 그것들을 리용하는 자료관리프로그람에 대하여 배운다. 그리고 부분프로그람으로서의 함수와 변수들의 준위개념 및 재귀적인 프로그람의 작성방법에 대하여 배운다.

다음으로 그림그리기프로그람작성에서 기초적인 지식들인 콤퓨터화면의 영상방식과 자리표계, 색의 표현방식에 대한 개념과 간단한 도형들과 움직이는 그림그리기프로그람작성방법에 대하여 배우게 된다.

모든것은 기초가 든든해야 훌륭한 결실을 볼수 있다.

우리는 이 과목학습을 실속있게 하여 앞으로 보다 능률적인 우리 식의 응용프로그람들을 개발할수 있는 기초를 튼튼히 다짐으로써 우리 학생들이 사회주의강성대국의 밝은 앞날을 떠메고나갈 훌륭한 과학기술인재로 자라나도록 크나큰 사랑과 은정을 베풀어주고 계시는 위대한 령도자 **김정일**원수님의 믿음과 기대에 높은 실력으로 보답하여야 한다.

제1장. 자료관리

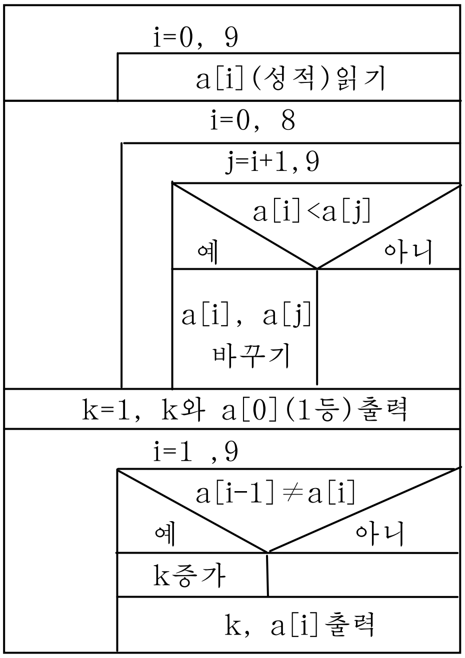
제1절. 성적순위내기

우리는 이 절에서 성적순위를 내는 문제를 푸는 과정을 통하여 배렬에 대하여 학습하기로 한다.

10명 학생들의 성적을 순위내여 출력하는 프로그람을 작성하자.

이 문제는 10명 학생들의 성적을 읽고 성적크기순서로 배렬한 다음 첫 학생부터 1등, 다음 학생은 2등, 이런 식으로 성적이 같은 학생은 같은 등수, 성적이 앞의 학생보다 작으면 다음 등수로 출력하는 문제이다. 알고리듬은 다음과 같다.

<알고리듬>



알고리듬에서는 10명 학생의 수학성적을 수학의 수렬에서와 같이 a0—a9로 표시하려고 하였다. 콤퓨터에서는 이것을 a[0]—a[9]로 표시한다.

순위를 내기 위하여 먼저 10명 학생의 성적을 읽고 성적을 크기순서로 배렬한다.

크기순서로 배렬하였을 때 첫 학생은 무조건 1등이므로 먼저 등수를 나타내는 변수 k에 1을 넣고 k와 첫 학생의 성적 a[0]을 출력한다. 다음 1번 학생부터는 자기의 앞성적(a[i]의 앞성적은 a[i-1])과 비교하여 같으면 앞학생과 같은 등수이므로 등수

k와 성적 a[i]를 그대로 출력하고 서로 다르면 앞학생과 서로 다른 등수로 되므로 등수 k를 하나 증가시키고 등수와 성적을 출력한다.

이 알고리듬에서 보는것처럼 번호에 따라 다른 자료들을 하나의 변수에 첨수를 붙여 표시한 a[0], a[1], …, a[9]를 배렬이라고 하고 프로그람에서 배렬로 관리한다. 그러므로 이 절에서는 배렬을 학습하면서 이 문제를 풀기로 한다.

1. 배렬변수의 정의

일상적으로 우리는 《정렬하다》, 《정돈하다》, 《배렬하다》라는 말을 많이 한다. 《책상을 높이순서로 배렬하라.》등과 같이 배렬이라는 말의 의미는 어떤 정해놓은 기준에 따라 정돈하여놓는다는것이다. 이때 정돈하여놓은 상태를 보면 같은 물건들이 차례번호를 가지고 라렬된다.

그러면 프로그람에서 배렬이란 무엇인가?

례로 우리 학급 학생들의 수학성적자료를 써놓으면 다음과 같다고 하자.

5, 4.8, 3.7, 4.2, 4.6, …, 5

이 성적들에서 공통점은 매 성적들의 자료형이 다 실수형이라는것이다. 이 성적들을 프로그람적으로 처리하려면 매 성적을 다 변수로 표시하여야 한다.

A=5, B=4.8, C=3.7, D=4.2, E=4.6, …

성적이 만일 100개라면 변수를 100개 써야 한다는것이다. 그런데 이렇게 서로 다른 변수를 100개 처리하려면 조건판정이나 읽기명령문만 해도 100개의 서로 다른 명령문을 리용하여야 한다.

이 100개의 서로 다른 성적을 한 변수로 표시한다면 하나의 명령문을 순환만 시켜 처리를 쉽게 할수 있을것이다.

이렇게 하는 방도가 바로 자료들을 배렬변수로 표시하는것이다.

배렬형이란 같은 형의 자료들로 이루어진 자료형이다.

우의 같은 형의 실수형자료 100개의 성적을 한 변수 a로 표시하고 100개의 성적을 다음과 같은 변수들로 표시한다.

a_1 , a_2 ,a_3 ,…, a_{100}

이때 a를 배렬변수라고 한다. 변수옆에 붙은 수자들을 첨수라고 한다.

이렇게 배렬형자료를 표시하는 첨수가 붙은 변수를 배렬변수라고 한다.

우의 배렬변수는 다음과 같이 선언하여야 리용할수 있다.

Float a[100];

배렬변수를 프로그람에서 리용하려면 선언을 하여야 한다.

2. 배렬변수의 선언

배렬형의 자료는 기억기의 배렬변수를 통하여 관리된다.

그러므로 배렬형자료를 프로그람에서 리용하려면 배렬형자료를 기억할 배렬형변수를 선언하여야 한다.

배렬변수의 선언형식:

자료형 배렬이름[요소수];

여기서 자료형은 배렬의 매 요소가 가지는 자료형이며 배렬이름은 배렬전체를 나타내는 변수이름이다.

요소수는 배렬요소의 최대개수이다.

이때 요소수에 따르는 첨수범위는 0∼(요소수-1)범위의 옹근수값으로 지정한다.

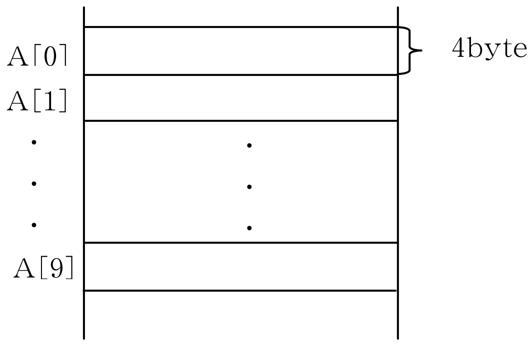
배렬이름은 변수이름만들기규칙에 따라 짓는다.

요소수는 배렬의 크기로서 옹근수형의 결과를 주는 산수식이여야 한다.

례 1:

```
int n=10;
    float A[n];
```

배렬선언을 하면 프로그람을 번역할 때 그 변수에는 다음과 같이 기억기가 할당된다.



그러므로 이때 할당되는 총 기억용량은 다음과 같이 계산된다.

총 기억용량=배렬요소 한개가 차지하는 기억용량×요소수

따라서 우의 례에서 총 기억용량=4byte×10=40byte

배렬을 잘 리용하려면 문제에서 자료들을 보고 배렬을 리용해야 하겠는가 안해도 되는가를 잘 판단하여야 한다.

배렬을 리용하려면 다음과 같이 준비를 잘해야 한다.

· 배렬의 매 요소가 무슨 형인가 판단하여야 한다.

례 2: 145.3, 112.6, 125.2, 150.6,… 이라면 float형
12, 15, 21, 15, 13, …이라면 int형을 선택하여야 한다.

· 배렬의 이름을 정한다.
배렬의 이름은 변수이름만들기규칙에 따라 뜻을 부여하여 정한다.
· 요소수를 결정한다.
배렬에 속한 자료수의 개수를 보고 정한다.

례 3: 20명 학생의 수학성적이라면 다음과 같이 선언할수 있다.

float math[20];

배렬변수선언에서 주의할 점은 다음과 같다.

· 요소의 자료형은 수값범위나 바이트수에서 배렬의 매 요소를 담을만 한 기억크기를 가지는 자료형이여야 한다.
· 요소수는 배렬에 속한 자료들을 다 기억시킬만큼 충분히 잡아야 한다.



탐 구

요소수를 지나치게 크게 정할 때와 적게 정할 때 어떤 결과가 나타나겠는가?

3. 배렬의 리용

선언된 배렬변수를 리용하려면 배렬의 매 요소를 어떻게 지정하는가를 알아야 한다.

배렬의 매 요소는 다음과 같이 지정한다.

배렬이름[첨수]

첨수는 0부터 요소수-1(요소수는 배렬선언시 취한 요소수)까지의 옹근수값이나 옹근수결과를 주는 산수식이여야 한다.

례 1: float a[10];

이와 같이 배렬을 선언하였을 때 배렬의 요소는 a[0]—a[9]와 같이 지정할수 있다.

례 2: 10명 학생의 콤퓨터성적자료를 읽어들이고 최우등생의 성적을 출력하는 프로그람을 작성하여라.

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
  float a[10];
  int i;
  for(i=1;i<=10;i++)
   scanf("%f", &a[i]);
  for(i=1;i<=10;i++)
   if(a[i]>=4.5) printf("%f\n", a[i]);
}
```

례 3: 10명 학생의 콤퓨터성적자료를 읽어들이고 성적이 제일 높은 학생의 자료를 출력하는 프로그람을 작성하여라.

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
 float a[10], max=0;
 int i;
 for(i=0;i<=9;i++)
  scanf("%f", &a[i]);
 max=a[0];
 for(i=0;i<=9;i++)
  if(max<a[i]) max=a[i];
 printf("max=%f\n", max);
}
```

례 4: 10명 학생의 성적을 크기순서로 배렬하는 프로그람을 작성하여라.

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
```

```
{
  float a[10],m=0;
  int i,j;
 for(i=0;i<=9;i++)
  scanf("%f", &a[i]);
  for(i=0;i<=8;i++)
   for(j=i+1;j<=9;j++)
   if(a[i]<=a[j])
    {
     m=a[i];a[i]=a[j];a[j]=m;
     }
   for(i=0;i<=9;i++)
    printf("%f\n", a[i]);
}
```

이 례까지 리용하면 우리가 이미 설정한 문제에 대한 코드는 다음과 같다.

<코드>

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
  float a[10],m=0;
  int i,j,k;
  for(i=0;i<=9;i++)
   scanf("%f", &a[i]);
  for(i=0;i<=8;i++)
   {
    for(j=i+1;j<=9;j++)
    if(a[i]<a[j])
     {m=a[i];
      a[i]=a[j];
      a[j]=m;
      }
    }
   k=1;printf("%d %f\n",k,a[0]);
   for(i=1;i<=9;i++)
   {
    if(a[i-1]!=a[i]) k++;
```

```
            printf("%d %f\n", k,a[i]);
            }
        }
```

해 설

프로그람에서 보면 10명 학생의 성적에서 i번째 학생성적을 a[i]로 표시하였으며 a[0]~a[9]은 0번 학생부터 9번 학생의 성적이다.

이것을 For순환을 리용하여 처리하도록 하였다.

main함수의 ⑤—⑥행코드는 a[0]—a[9]성적을 읽어들이는 코드이다.

⑦—⑮은 a[0]—a[9]을 크기순서로 배렬하는 코드이다.

그뒤의 코드는 순위와 성적을 출력하는 코드이다.

례에서 보는바와 같이 10명 학생의 수학성적과 같이 같은 형의 자료들의 렬을 a[0]—a[9]과 같이 하나의 변수 a를 통하여 표시하고 리용하였다.

4. 1차원배렬과 2차원배렬

우리가 이미 10명 학생의 성적을 배렬로 다음과 같이 선언하였다.

```
Float a[10];
```

이 배렬의 매 요소를 a[0], a[1], …, a[9]와 같이 표시하였다.

여기서 배렬변수 a에는 첨수가 1개씩 붙어있는데 이 첨수가 0부터 9까지 변하였다.

첨수가 1개인 배렬을 1차원배렬이라고 한다.

첨수가 2개인 배렬을 2차원배렬이라고 한다.

우리가 이미 학습한 배렬들은 바로 1차원배렬들이였다.

10명 학생의 수학성적에서는 배렬변수 a를 특징짓는 량이 학생번호 하나였다.

그러나 례로 10명 학생의 세 과목성적표가 주어졌다고 하자.

번호	수학	물리	콤퓨터
1	4.8	4.7	4.3
2	4.3	4.5	4.7
…	…	…	…
10	4.5	4.1	4.6

이 표에서 보는바와 같이 하나의 성적을 특징짓는 량은 학생번호(1—10)와 과목번호(1—3)로서 두가지 량이다.

이때 4.8은 1번학생의 수학과목성적, 4.7은 1번학생의 물리과목성적, …, 4.6은 10번학생의 콤퓨터과목성적이라고 말할수 있다.

그러면서도 매 성적은 다 실수형자료로서 자료형이 다 같다.

이 성적자료를 배렬변수 a라고 표시한다면 매개 요소자료들은 다음과 같이 표시할 수 있다.

1번학생의 1번과목성적 — a_{11}
1번학생의 2번과목성적 — a_{12}
… …
10번학생의 3번과목성적 — $a_{10\ 3}$

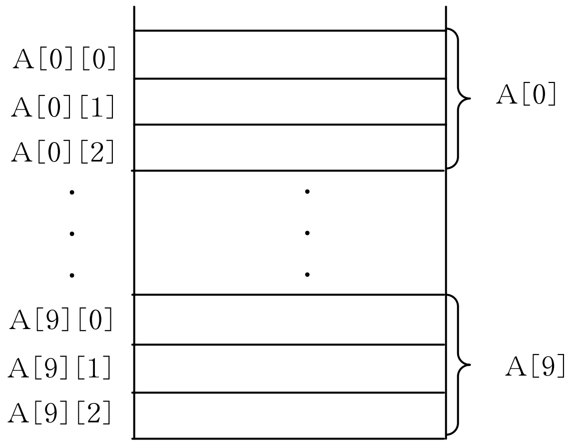
여기서 보는것처럼 배렬변수 a에는 첨수가 다 2개씩 붙어있다.
이때의 배렬을 2차원배렬이라고 한다.

2차원배렬의 선언형식:

자료형 변수이름[요소수1][요소수2];
요소수1: 첫째 첨수의 요소의 최대개수
요소수2: 둘째 첨수의 요소의 최대개수

례 1: float a[10][3];

이렇게 선언하면 기억기에는 다음과 같이 배렬변수가 할당된다.



2차원배렬변수의 기억바이트수는 다음과 같다.

총 기억바이트수=요소의 자료형이 차지하는 기억바이트수×요소수1×요소수2

3차원배렬, 4차원배렬의 경우도 마찬가지이다.
어떤 차원수의 배렬이라고 해도 배렬의 선언과 기억바이트수구하기는 마찬가지이다.
배렬의 리용에서 주의할 점은 다음과 같다.

· 배렬의 초기값주기가 있다면 요소수를 생략할수 있다. 그러면 요소수는 초기값의 개수로 된다.

그러나 초기값을 주지 않으면 생략할수 없다.

```
례 2: int a[]={1, 2, 3}; // 배렬의 초기값주기
     int a[]; // 오유
```

· 초기값주기에서 초기값의 개수가 배렬의 요소수보다 작으면 나머지 요소에는 0이 채워지며 많으면 오유로 된다.

```
례 3: int a[5]={20, 50, 70};  경우
      a[0]=20, a[1]=50, a[2]=70, a[3]=0, a[4]=0으로 된다.
```

· 배렬의 초기값주기를 선언부가 아닌 실행부에서는 절대로 할수 없다.

```
례 4: int a[5];
      a={10,30,50,20,60}; // 오유
```

· 배렬에 대한 값주기에서 첨수의 한계를 넘는 요소에 대한 값주기는 절대로 할수 없다. 첨수의 한계를 넘는 요소에 대한 처리를 진행하면 오유통보를 발생한다.

```
례 5: int a[5];
      a[6]=3;// 오유
```

· 프로그람에서 배렬의 값주기를 하려면 반드시 순환문을 리용하여 차례로 요소별로 값주기를 하여야 한다.

```
례 6: int a[4];
      int i;
      For(i=0; i<4; i++)
       scanf("%d",&a[i]);
 또는
     int a[4];
     int i;
     for(i=0;i<4;i++)
       a[i]=i*2;
```

· 배렬변수의 이름에 절대로 값주기를 할수 없다.

```
례 7: int a[7];
     int b[7];
     a=3;// 오유

례 8: int a[3]={3, 10, 20};
     int b[3];
      b=a;// 오유
```

우의 코드를 옳바로 작성하면 다음과 같다.

```
int b[3],a[3]={3, 10, 20};
int i;
for(i=0;i<4;i++)
 b[i]=a[i];
```



탐 구

배렬에서 2차원배렬의 원소들을 처리하려면 무조건순환명령문을 몇개 써야 하는가?

다차원배렬이란 차원수가 2이상인 배렬을 말한다.

레 9: 5명 학생의 3과목 성적이 주어졌다. 매 학생의 학생별평균성적을 출력하는 프로그람을 작성하여라.

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
  float a[5][3],s;
  int i,j;
  for(i=0;i<=4;i++)
   for(j=0;j<=2;j++)
   scanf("%f", &a[i][j]);
  for(i=0;i<=4;i++)
   {s=0;
    for(j=0;j<=2;j++)
      s+=a[i][j];
    printf("%f\n", s/3);
   }
  }
```

련습문제

1. 20명 학생들의 키측정자료가 있다. 짝수번째와 홀수번째 학생들의 평균키를 출력하는 프로그람을 작성하여라.
2. n개의 실수가 주어졌을 때 소수부가 큰 순서로 출력하여라.
3. 20명 학생의 3과목 성적이 주어졌다. 과목별평균성적을 출력하는 프로그람을 작성하여라.
4. 10개의 분수가 주어졌다. 크기순서로 배렬하여 분수형태로 출력하여라.
5. 10명 학생의 콤퓨터시험성적이 주어졌다. 평점보다 높은 학생들의 인원수를 출력하는 프로그람을 작성하여라.
6. 20명 학생들의 3과목 성적이 주어졌다. 학생별평점순위로 매 학생들의 자료를 출력하여라.
7. 5개 학급 학생들의 영어본문통달경연성적이 주어졌다. 학급별평점과 순위를 출력하여라.
8. 10진수를 2진수로 변환하는 프로그람을 작성하여라.(배렬을 리용하여라.)
9. 합이 1 994인 두 자연수들가운데서 그들사이의 적이 최대로 되는 두 자연수를 출력하여라.
10. 5명 학생의 3과목 성적이 주어졌다. 전과목 최우등생인 학생수를 출력하는 프로그람을 작성하여라.
11. n개의 2차함수가 주어졌을 때 구간 [x1, x2]에서 가장 큰 값을 가지는 함수를 출력하여라.





콤퓨터상식

부단히 진화하는 인공지능의 과제

1960년대에는 오늘날의 《붉은별》이나 《Linux》등의 콤퓨터조작체계와 같은 《편리》한 사람-콤퓨터사이의 대면부를 잘 실현하는것이 인공지능의 연구과제였다. 콤파일러나 프로그람언어도 인공지능의 대상으로 되였었다.

1970년대에는 지식을 사용하여 추론하는것 등이 인공지능의 대상으로 되였으며 극히 얼마전까지는 장기와 같은 게임이 인공지능의 연구과제였으나 지금에 와서는 달라졌다. 인공지능은 일단 연구목표가 실현되기만 하면 그 이후부터는 그 연구결과가 사람의 일상생활에서 《편리한 도구》로 되여버리며 이렇게 되면 인공지능연구의 대상으로부터 탈퇴하여 일상적인 정보처리로 되고만다.

인공지능이란 인간에게 고유한 기능이나 지능이라고 생각되는것을 콤퓨터를 사용하여 실현하는 공학, 기술 및 그것을 연구하는 과학이다. 현재까지 인공지능연구는 《튜링시험》에 의하여 정의되고 평가되는 기준에 따라 착실하게 발전하여왔다고 말할수 있다. 그러나 만족을 모르는 인공지능전문연구자들은 튜링시험만으로는 불충분하며 그 알속을 조사하지 않고서는 인공지능이라고 말할수 없다는 립장을 취하고있다. 이런 경우는《인간자신을 조사하는 수단》으로서의 인공지능연구이다. 이런 높은 기준에서 본다면 인공지능실현에로의 길은 멀고도 멀다. 앞으로 인공지능은 생명, 진화 그리고 자체개변하는 프로그람과 융합하는 장치 등에서 새로운 발전이 이룩될것으로 보아지고있다.

제2절. 문장에서 단어찾기

1. 문자렬의 정의

콤퓨터에서는 하나이상의 문자들의 묶음으로 이루어진 자료를 문자자료라고 한다.

C언어에서는 문자렬을 문자의 1차원배렬로 취급한다.

다시말하여 문자렬은 그 문자렬을 구성하는 매개 문자들의 배렬로 취급된다.(문자렬형이라는 자료형은 없다.)

자료형이 Char형인 배렬을 문자배렬 간단히 문자렬이라고 한다.

문자렬의 선언형식:

```
char 배렬이름[요소수];
```

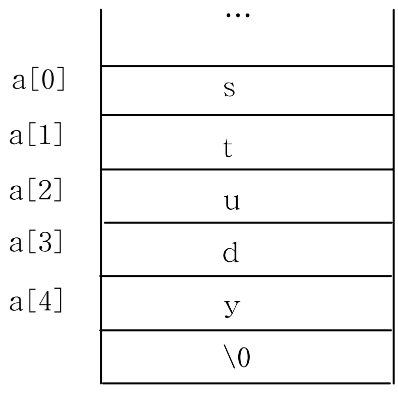
례 1: char string[10];// 9개의 문자를 가진 문자렬을 보관할수 있는 배렬변수에 대한 선언

배렬이라는 측면에서 볼 때 문자렬에서는 수값배렬과는 달리 배렬의 맨 마지막원소가 \0으로 끝나게 되여있다.

그 원인은 문자렬이 끝나면 다른 문자렬과 구별되여야 하므로 체계에 의하여 문자렬끝을 의미하는 문자렬끝기호인 \0기호를 자동적으로 붙이게 된다.

례 2: char string[5]= “study”;

이와 같이 선언된 경우 기억기에는 문자렬이 다음과 같이 할당된다.



기억기가 할당된것을 보면 문자렬끝에 령문자 \0가 붙게 되므로 문자렬을 배렬로 선언할 때는 요소수를 자기길이보다 하나 더 크게 해주어야 한다.

문자렬도 배렬이므로 선언시 초기화를 할수 있다.

례 3: char s[12]= “my computer”

이 경우 배렬 s의 요소들은 프로그람에서 다음과 같이 값주기한것과 같다.

s[0]=“m”, s[1]=“y”, s[2]=“ ”, s[3]=“c”, s[4]=“o”, s[5]=“m”, s[6]=“p”, s[7]=“u”, s[8]=“t”, s[9]=“e”, s[10]=“r”, s[11]=“\0”

C언어에서 취급하는 모든 문자렬은 아스키문자렬이다.

례 4: 5명 학생의 이름을 입력하여 성이 kim인 학생수를 구하는 프로그람을 작성하여라.

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
 char s[5][10];
 int i,k;
 for(i=0;i<=4;i++)
  scanf("%s", &s[i]);
 for(i=0;i<=4;i++)
  if(strncmp(s[i],"kim",3)==0) k++;
 printf("%d\n", k);
}
```

례 5: 문자렬을 입력하여 이 문자렬을 이루고있는 매개 문자의 아스키코드를 출력하는 프로그람을 작성하여라.

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
  char s[5];
  int i;
  scanf("%s", &s);
  for(i=0;i<=4;i++)
   printf("%d\n", s[i]);
}
```

2. 문자렬표준함수

1) 문자렬입력함수

문자렬입력에는 함수 scanf와 gets를 리용한다.

scanf로 입력할 때에는 문자렬변수의 앞에 &기호를 붙이지 않는다.

gets함수의 형식:

```
gets(문자렬변수이름);
```

기능:

지적된 문자렬변수이름에 문자렬이 입력된다.

례 1: char s[10];
gets(s);

프로그람에서 gets함수와 scanf함수를 함께 쓰면 함수의 호출이 서로 영향을 미치게 된다. 이 영향을 막자면 다음과 같은 명령문을 추가하면 된다.

```
fflush(stdin);
```

2) 문자렬출력함수

문자렬의 출력은 함수 printf(), puts(), cprintf()를 리용한다.

3) 문자렬비교함수

함수의 형식:

```
strcmp(문자렬1, 문자렬2);
```

기능:

두 문자렬의 크기를 비교한다. 만일 문자렬1이 문자렬2보다 크다면 함수는 정의 옹근수를 돌려주고 문자렬1이 문자렬2보다 작다면 함수는 부의 옹근수를 돌려준다. 그리고 두 문자렬이 같으면 함수는 0을 돌려준다.

문자렬의 비교규칙
두 문자렬의 왼쪽으로부터 오른쪽으로 가면서 하나씩 아스키코드값을 비교하면서 서로 다른 문자가 나타나거나 \0문자를 만나면 끝낸다. 이때 모든 문자가 같아야 두 문자렬이 같은것이다. 서로 다른 문자가 나타났을 때에는 처음으로 나타난 서로 다른 문자의 비교결과를 기준으로 한다.

례로 strcmp(“ACC”, “ABC”)의 결과는 정의 옹근수이며 strcmp(“ABC”, “ACC”)의 결과는 부의 옹근수이다. 그것은 두개의 문자렬 ABC와 ACC에서 두번째 문자부터 차이나는데 B의 아스키코드가 C의 아스키코드보다 작기때문이다.

례 2: 사용자가 암호를 입력하면 판정하고 옳으면 yes, 틀리면 no를 출력하는 프로그람을 작성하여라.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define password "myclass"

int main(int argc, char *argv[])
{
  char pwd[8];
  gets(pwd);
  if(strcmp(pwd,password)==0)
    printf("yes\n");
  else
    printf("no\n");
  }
```

4) 문자렬복사함수

함수의 형식:

strcpy(문자렬1, 문자렬2);

기능:

문자렬2를 문자렬1에 복사한다.(끝문자 \0도 포함) 함수의 첫 인수는 반드시 문자형배렬변수이여야 하며 두번째 인수는 문자렬을 포함하고있는 문자형배렬변수이거나 문자렬상수일수도 있다.

례 3:
```
char s1[10], s2[10];
  scanf("%s",s2);
  strcpy(s1,s2);
```

문자렬의 일부만을 복사하려면 strncpy()함수를 리용하여야 한다.

함수의 형식:

strncpy(문자렬1, 문자렬2, 길이n);

기능:

문자렬2앞의 n개의 문자를 문자렬1에 복사하고 끝에 \0을 덧붙인다.

례 4:
```
char s[10],s[ ]= "ABC";
  strncpy(s1,s2,2);
  printf("%s",s1);
```

이 함수의 리용에서 문자렬1은 문자렬2를 포함할수 있어야 한다. 그렇지 않으면 한계넘침조작이 일어나므로 위험하다.

5) 문자렬련결함수

함수의 형식:

```
strcat(문자렬1, 문자렬2);
```

기능:

문자렬2를 문자렬1의 문자렬뒤에 련결한다.

례 5: char s1[]= “my ”, s2[]= “computer”;

```
strcat(s1,s2);
printf(“%s”, s1);
```

이 코드의 결과는 my computer이다.

6) 문자렬길이구하기함수

함수의 형식:

```
strlen(문자렬);
```

기능:

문자렬의 길이를 돌려준다.

례 6: char s[]= “computer”;

```
printf(“%d”,strlen(s));
```

실행결과는 8이다.

례 7: 한개의 문장을 입력하고 이 문장에 있는 단어의 개수를 출력하는 프로그람을 작성하여라.

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
  char string[30];
  int i,num=0,mark=0;
  char c;
  gets(string);
  for(i=0;string[i]!='\0';i++)
    {
```

```
      c=string[i];
     if(c==' ')
       mark=0;
     else if(mark==0)
      {
       mark=1;
      num++;
      }
     }

    printf("words=%d\n", num);
}
```

련습문제

1. 10개의 문자렬자료가 주어졌다. 문자렬자료들가운데서 첫 글자가 A로 시작되는 문자렬의 개수를 출력하는 프로그람을 작성하여라.
2. 2진수를 16진수로 변환하는 프로그람을 작성하여라.
3. 2진수를 8진수로 변환하는 프로그람을 작성하여라.
4. 16진수를 2진수로 변환하는 프로그람을 작성하여라.
5. 8진수를 2진수로 변환하는 프로그람을 작성하여라.
6. 10진수를 8진수로 변환하는 프로그람을 작성하여라.
7. 10진수를 16진수로 변환하는 프로그람을 작성하여라.
8. 한개의 문장을 입력하였다. 이가운데서 computer라는 문자렬이 몇개 있는가를 출력하는 프로그람을 작성하여라.
9. 한개의 문장을 입력하였다. 여기서 매 문자의 빈도수를 출력하는 프로그람을 작성하여라.
10. 한개의 문장을 입력하였다. 매 단어의 빈도수를 출력하는 프로그람을 작성하여라.
11. 10개의 문자렬이 주어졌다. 아스키코드순서로 문자렬을 배렬하는 프로그람을 작성하여라.
12. 한개의 문장을 입력하면 매 문자의 아스키코드를 출력하는 프로그람을 작성하여라.
13. 문장을 입력하였다. 이가운데서 공백을 제거한 문자렬을 출력하는 프로그람을 작성하여라.
14. 문자를 계속 입력한다. #건이 눌리울 때까지 눌리운 건의 개수와 매 문자의 빈도수를 출력하여라.
15. 20개의 문자를 입력하였다. 이가운데서 영어문자의 개수, 수자의 개수, 기타 기호의 개수를 출력하는 프로그람을 작성하여라.





콤 퓨 터 상 식

쏘프트웨어의 판번호는 무엇을 의미하는가

쏘프트웨어제품들은 단번에 완성되는것이 아니다.

쏘프트웨어의 판번호는 그것의 발전완성단계를 표시한다. 즉 작성된 어떤 쏘프트웨어의 리용과정에 나타난 불량상태나 오유들을 수정하거나 쏘프트웨어자체의 개선을 위한 변경을 진행한것을 나타내는 표식이다.

이때 그 번호를 1.0, 1.1, 1.52, 2.0 등과 같이 붙이는데 이것은 매 체계나 프로그람의 특징을 나타내는것이다.

일반적으로 앞의 정수부분이 바뀌면(례로 1.0에서 2.0으로) 프로그람자체의 구조나 기능이 대폭 수정되고 보충되였다는것을 의미한다.

그리고 첫번째 소수부가 바뀌면(례로 1.0에서 1.1로) 프로그람이나 체계의 한 부분이 약간 수정보충되였다는것을 의미한다.

두번째 소수부가 바뀌면(례로 1.51에서 1.52로) 프로그람이나 체계구성상 아주 미세한 부분이 수정되였다는것을 의미한다.

제3절. 학생자료검색

우리는 이미 체계가 정의하고있는 표준자료형들에 대하여 학습하였다.

그러나 표준자료형만 가지고서는 현실에서 제기되는 모든 문제들을 프로그람적으로 처리하기 힘들다. 그러므로 2절에서는 같은 형의 자료들을 하나의 변수로 묶어 관리할수 있는 배렬에 대하여 학습하였다. 현실에서는 같은 형의 자료들만 제기되는것이 아니라 서로 다른 자료형으로 이루어진 자료들이 더 많이 제기된다.

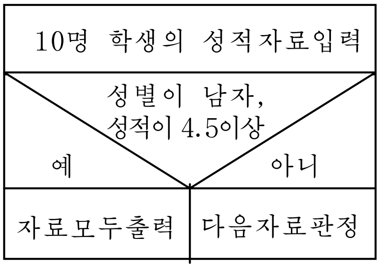
이 절에서는 배렬과 달리 서로 다른 여러가지 자료들을 묶어 처리할수 있는 구조체형에 대하여 학습하게 된다.

C언어는 표준자료형외에 사용자가 정의하고 리용할수 있는 사용자정의형이 있다.

사용자정의형에는 구조체형과 공용체형, 렬거형이 있다. 여기서는 학생자료검색을 위한 문제를 해결하는 과정을 통하여 프로그람작성에서 많이 쓰는 구조체형에 대하여 학습하기로 한다.

10명 학생들의 이름, 성별, 나이, 성적이 있다. 이 학생들가운데서 최우등생인 남학생들의 자료를 검색하여 출력하는 프로그람을 작성하자. 이 문제에서는 한 학생의 자료를 이루는 매개 요소자료들 즉 이름, 성별, 나이, 성적들의 자료형이 서로 다른데 학생별로 처리해야 하므로 매 학생의 요소자료들을 묶을수 있는 자료형을 만들면 다루기 편리하고 쉽게 처리할수 있을것이다. 알고리듬은 다음과 같다.

<알고리듬>



알고리듬적방법은 우리가 이미 다른 자료들에서 적용하였던것이다. 여기서 기본은 서로 다른 자료형들로 이루어진 매 학생에 대한 자료형을 만들고 이것을 어떻게 합리적으로 리용하겠는가 하는것이다.

이렇게 서로 다른 자료형으로 이루어진 구조체에 대한 정의와 리용방법을 보기로 하자.

1. 구조체형자료의 정의

구조체형이란 서로 다른 몇개의 련관이 있는 자료를 하나로 묶어 취급하는 자료형이다.

학생명단에서 학생이름은 문자렬형, 성별은 문자렬형, 나이는 옹근수형, 성적은 실수형, 집주소는 문자렬형, 키와 몸질량은 실수형으로서 한 학생의 자료를 이루고있는 매 요소자료형들이 서로 다르다. 이러한 자료형이 서로 다른 몇개의 자료들을 묶어 한 학생에 대한 자료를 정의하는 자료형이 바로 구조체형이다.

구조체형의 정의형식:

```
struct <구조체형이름>
 {
     형이름1   요소이름1;
     형이름2   요소이름2;
     형이름3   요소이름3;
       …           …
     형이름n   요소이름n;
 } ;
```

여기서 구조체형이름은 사용자가 새로 정의하는 구조체형을 식별하기 위한 이름이다.

이렇게 정의한 자료형이름은 프로그람안에서 체계가 정의한 표준자료형들인 char, int, short, long 등과 꼭같이 작용한다.

구조체형이름도 C언어의 이름짓기규칙에 따라 정의해야 한다.

형이름들은 구조체를 이루는 매개 요소들의 자료형이며 요소이름들은 구조체를 이루는 매개의 서로 다른 자료들을 대신하여 사용자가 만든 변수이름들이다. 그러므로 이 변수이름들도 이름만들기규칙에 따라 만들어야 한다.

례 1: 학생의 번호, 이름, 성별, 나이, 콤퓨터성적, 집주소를 가지고 구조체형을 정의하여라.

```
struct student
  {
    short num;
    char name[10];
    char sex;
    short age;
    float mark;
    char address[15];
  };
```

례 2: 도서관의 책목록에서 매 책에 대한 자료를 가지는 구조체형을 정의하여라. 책목록에는 책의 제목, 페지수, 출판년도, 출판사이름, 집필자이름이 있다.

```
struct book
  {
    char name[20];
    long page;
    long year;
    char publisher[20];
    char author[10];
  };
```

2. 구조체형변수의 선언

구조체형변수의 선언방법에는 두가지가 있다.

① 구조체형을 정의하고 변수이름을 따로 선언하는 방법이다.

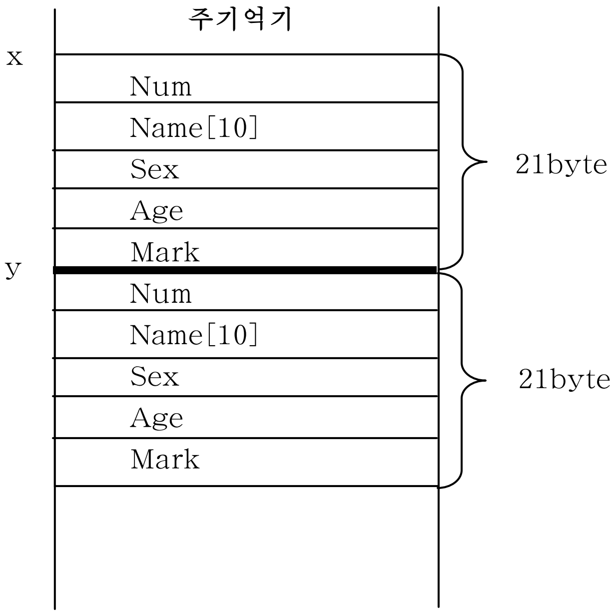
struct <구조체형이름> <변수이름1>, <변수이름2>,…, <변수이름n>;

례:
```
struct student
  {
    short num;
    char name[10];
```

```
        char sex;
        short age;
        float mark;
        };
    struct student x,y;
```

이렇게 변수선언을 진행하면 프로그람번역시 콤퓨터는 주기억기에 구조체형 student가 가지고있는 요소들의 기억바이트수의 총합과 같은 기억바이트수를 배당하고 이것의 주소를 각각 x, y에 준다.

구조체형은 바로 구조체형변수가 선언된 다음에야 기억기배당이 진행된다.



② 구조체형의 정의와 구조체형변수선언을 하나로 묶어서 진행하는 방법이다. 우의 례를 이 방법으로 선언하면 다음과 같다.

```
      struct student
        {
           short num;
          char name[10];
          char sex;
          short age;
          float mark;
        } x, y;
```

3. 구조체형변수의 리용

선언된 구조체형변수는 다음과 같이 리용한다.

리용형식:

구조체형변수이름.요소이름

구조체형변수는 자기의 자료형안에 서로 다른 요소들을 또 가지고있으므로 프로그람에서 변수의 구체적인 요소를 지적하여야 한다.

례 1: struct student

```
{
   short num;
   char name[10];
   char sex;
   short age;
   float mark;
 };
 struct student x,y;
 scanf("%d%s%s%d%f%s",&x.num, &x.name[0], &x.sex[0], &x.age,
                                         &x.mark, &x.address[0]);
 y.num=1;
 y.name="LiYongNum";
 y.sex="M";
 y.age=15;
 y.mark=4.6;
 y.address="BoTong river district";
 printf("%d%s%s%d%f%s",x.num, x.name, x.sex, x.age, x.mark,
                                                    x.address);
 printf("%d%s%s%d%f%s",y.num, y.name, y.sex, y.age, y.mark,
                                                    y.address);
```

우의 내용을 종합하여 앞에서 제기한 문제의 코드를 작성하려면 먼저 서로 다른 자료들로 이루어진 한 학생의 자료를 나타내는 자료형인 구조체 student를 만들고 이것을 리용하여 10명 학생의 요소자료들에 대한 검색을 진행하는 코드를 작성한다.

<코드>

```
include <stdio.h>
main()
  {
struct student
  {
     short num;
     char name[10];
     char sex;
     short age;
     float mark;
   };
struct student x[10];
int i;
for(i=0;i<10;i++)
  scanf("%d%s%s%d%f",&x[i].num, &x[i].name[0], &x[i].sex[0], &x
        [i].age, &x[i].mark);
for(i=0;i<10;i++)
if(x[i].sex= 'm' && x[i].mark>=4.5)
  printf("%d%s%s%d%f",x[i].num, x[i].name[0], x[i].sex[0], &x[i].
        age, x[i].mark);
}
```

코드에서 보는바와 같이 구조체 student를 만들어 리용함으로 하여 한 학생안에서도 서로 다른 형의 자료들을 따로 취급할 복잡성을 피하고 학생별로 자료처리를 하기가 편리하게 되여있다.

구조체형안의 요소가 또 구조체형일수도 있다.

례 2: 우의 례에서 나이가 아니라 난날이라면 난날을 규정하는 구조체형을 만들고 다시 학생자료를 정의하는 구조체형안에서 이 난날형을 리용할수 있다.

```
struct date
   {
     short year;
     short month;
     short day;
    };
```

```
struct student
    {
      short num;
      char name[10];
      char sex;
      struct date birth;
      float mark;
    }x;
```

여기서 보는것처럼 난날을 나타내는 birth변수는 date구조체형의 구조체형변수로 되면서 student구조체형의 요소이름으로 된다.

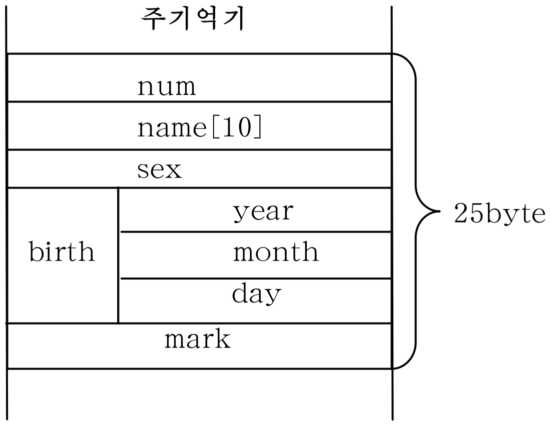
이렇게 구조체형변수의 요소가 다시 구조체형으로 된 경우에는 구조체형의 요소지정을 다음과 같이 진행한다.

<어미구조체형변수이름>.<요소구조체형변수이름>.<요소이름>

우의 례에서 x학생의 난날에서 년, 월, 일은 다음과 같이 값주기할수 있다.

```
x.birth.year=1995;
x.birth.month=3;
x.birth.day=27;
```

이와 같이 구조체형변수 x를 선언하면 기억구역은 다음의 그림과 같이 배당된다.



구조체형변수를 선언하면 프로그람의 번역이 진행될 때 기억공간이 구조체형만큼 확보된다. 이때 할당된 기억구역에는 예측할수 없는 임의의 자료가 들어있게 된다. 그것은 할당된 기억구역이 이미 전에 다른 응용프로그람에 의하여 리용되였기때문이다. 그러므로 프로그람의 혼란을 피하기 위하여 초기에 기억구역의 자료를 지우거나 어떤 일정한 값을 설정해줄 필요가 제기된다.

프로그람이 실행되기 전에 변수로 리용할 기억구역을 어떤 일정한 값으로 설정하는것을 변수의 초기화라고 한다.

구조체형변수도 초기화할수 있다.

구조체형변수의 초기화는 구조체형변수를 선언하는 동시에 괄호 { }를 리용하여 요소수만큼 초기값을 넣어주는 방법으로 진행한다.

례 3: 한 학생의 자료를 입력하여 출력하여라.

```
include <stdio.h>
main()
  {
struct student
  {
     short num;
     char name[10];
     char sex;
     short age;
     float mark;
   };
struct student x={1, “김영철”, “남”, 15, 4.5};
printf(“%d”,x.num);
printf(“%s”, x.name);
printf(“%s”,x.sex);
printf(“%f\n”,x.mark);
}
```

례 4: 번호, 이름, 성별, 나이, 성적으로 된 학생명단자료가 있다. 이가운데서 나이가 15살이상인 학생들의 이름을 출력하는 프로그람을 작성하여라.

```
include <stdio.h>
main()
  {
struct student
  {
     short num;
     char name[10];
     char sex;
     short age;
     float mark;
   };
```

```
struct student x[10];
int i;
for(i=0;i<10;i++)
  scanf("%d%s%s%d%f",&x[i].num, &x[i].name[0], &x[i].sex[0], &x
         [i].age, &x[i].mark);
for(i=0;i<10;i++)
if(x[i].age>=15)
 printf("%s", x[i].name);
}
```

례 5: 우의 문제에서 나이가 15살이상이면서 성적이 최우등인 학생들의 자료를 출력하여라.

```
include <stdio.h>
main()
  {
struct student
  {
     short num;
     char name[10];
     char sex;
     short age;
     float mark;
   };
struct student x[10];
int i;
for(i=0;i<10;i++)
 scanf("%d%s%s%d%f",&x[i].num, &x[i].name[0], &x[i].sex[0], &x
      [i].age, &x[i].mark);
for(i=0;i<10;i++)
if(x[i].age>=15 && x[i].mark>=4.5)
 printf("%d%s%s%d%f\n",x[i].num,x[i].name, x[i].sex, x[i].age, x
        [i].mark);
}
```

련습문제

1. 10명 학생의 이름, 성별, 성적이 주어졌다. 이가운데서 남학생의 평균성적과 녀학생의 평균성적을 출력하는 프로그람을 작성하여라.
2. 10명 학생의 이름, 성별, 나이가 주어졌다. 평균나이를 구하여 평균나이보다 나이가 많은 학생의 이름과 성별을 출력하여라.
3. 5명 학생의 이름, 키, 100m 달리기, 높이뛰기, 너비뛰기 판정결과가 주어졌다. 전체 종목에서 합격한 학생의 자료를 출력하여라.
4. 10명 학생의 이름, 수학, 물리, 콤퓨터성적이 주어졌다. 전과목 최우등생인 학생의 인원수를 출력하여라.
5. 15개의 점의 색갈과 자리표가 주어졌다. 1사분구, 2사분구, 3사분구, 4사분구에 놓이는 점들을 따로따로 갈라 자리표와 색갈을 출력하여라.
6. 우의 문제에서 색갈별로(색갈은 red, green, blue가 있다고 보아라.) 점의 개수를 출력하여라.
7. 어떤 상점의 상품번호, 품종, 들어온 날자, 단가, 상품량자료가 주어졌다. 매 상품별로 날자와 총 금액을 출력하여라.
8. 우의 문제에서 상품의 총 금액순위로 상품자료를 배렬하여라.





콤퓨터상식

콤퓨터에서 무정전전원장치는 왜 필요한가

무정전전원장치(UPS)를 사용하면 콤퓨터작업시 전원공급이 갑자기 끊어져도 일정한 시간동안은 정전상태가 되지 않는다.

일반적으로 콤퓨터사용시 갑자기 정전이 되면 콤퓨터에 기억되여있는 자료가 파괴될수 있으며 미처 작업내용을 보존하지 못하여 다시 작업을 하여야 한다. 보통 이러한 사정을 사람들은 피치 못할 사정으로 보고 대하는것이 일쑤이다. 그러나 콤퓨터에서의 정전을 절대로 허용하지 말아야 할 경우도 있다. 례를 들면 콤퓨터조종으로 환자를 치료하는 의료기구가 갑자기 정전이 되면 치료가 지연되게 되며 심지어는 환자의 생명이 위급해질수 있다.

하나의 전자식전원전압안정장치인 무정전전원장치는 교류입구전원의 변화가 한정범위를 초과하거나 갑자기 정전이 되였을 때 자체내에 있는 용량이 큰 축전지의 지원으로 자동적으로 콤퓨터에 에네르기를 공급해줌으로써 일정한 시간동안 콤퓨터에서 작업하던 자료를 재빨리 기억시키고 콤퓨터의 가동을 정상적으로 끝낼수 있는 시간적여유를 준다.

제4절. 지적자

1. 지적자의 정의

지적자란 말그대로 어떤 대상을 가리킨다는것이다.

콤퓨터에서 자료를 지적하자면 그 자료가 기억되여있는 주소를 알아야 한다.

지적자는 지적하려는 대상이 기억되여있는 기억기의 주소이다.

변수는 자료를 일시적으로 기억하기 위한 주소에 붙인 이름이다.

변수는 사용자가 변수의 구체적인 주소값은 몰라도 변수이름만 알면 그 내용을 리용할수 있다. 즉 변수를 리용하여 연산을 진행하면 주소가 아니라 그 주소에 등록된 값들만이 연산에 참가하게 된다.

그러나 지적자는 연산에 참가하는 변수의 주소를 관리할수 있게 하는 변수로서 주소조작을 진행한다. 다시말하여 변수는 기억기안에 들어있는 값을 직접 호출(접근)하는것이며 지적자는 변수의 주소를 보관하고 그를 통하여 기억기를 간접적으로 조작하는것이다. 이런 의미에서 주소를 인위적으로 줄수 없는 변수를 흔히 말하는 변수라고 하고 주소조작을 할수 있는 변수를 지적자라고 한다.

C언어에서는 주소를 보관할수 있는 변수의 형 즉 특별히 지적자형이라고 하는 자료형의 한 종류를 제공한다.

지적자형변수란 프로그람에서 전문적으로 어떤 자료형을 가진 변수의 주소값을 기억해두는 변수를 말한다. 지적자형변수를 간단히 지적자라고 하기로 한다.

우리가 이미 학습한 변수들은 지적자와 밀접한 련관관계가 있으며 지적자를 리용하면 자료들의 처리를 보다 쉽게 간단히 할수 있다.

2. 배렬과 지적자

배렬은 이름이 같고 형이 같은 자료들의 묶음으로서 묶음의 매 요소들이 기억기에 순차적으로 기억되여있으므로 매 요소를 첨수에 의하여 호출한다.

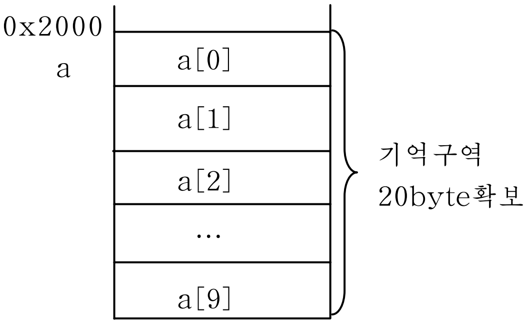
배렬의 이러한 특성으로부터 배렬의 첫 주소만 알면 매 요소는 자료형의 크기에 따라 주소를 증가시키거나 감소시키는 방법으로 호출할수 있다. 즉 지적자와 배렬은 매우 밀접한 관계를 가진다. 따라서 배렬은 첨수에 의하여 관리하던 요소들을 지적자를 써서 편리하게 실현할수 있다.

배렬변수는 그대로 지적자변수로 된다.

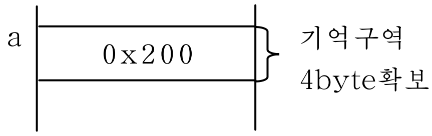
례로 int a[10]이라고 변수선언을 진행하는것과 int *a라고 선언하는것은 같은 배렬선언으로 된다.

차이점은 배렬선언에서는 int형변수 10개에 대한 기억구역을 확보하지만 지적자로 선언하면 배렬의 첫 주소를 보관하기 위한 기억구역만 확보된다는 면에서 서로 다르다.

int a[10]로 선언된 경우



int *a로 선언된 경우



례: 10명 학생의 수학성적배렬을 선언하고 성적을 입력한 다음 지적자를 써서 짝수번째 학생들의 합과 홀수번째 학생들의 합을 구하는 프로그람을 작성하여라.

```
#include <stdio.h>
void main(void)
{
int a[10];
int *p, s1,s2,I;
 s1=s2=0;
 for(i=0; i<10;i++)
  scanf("%d", &a[i]);
  p=x;
 for(i=0;i<5;i+=2)
  {
     s1=s1+*(p+i);
     s2=s2+*(p+i+1);
   }
     printf("s1=%d\n",s1);
     printf("s2=%d\n",s2);
}
```

해 설

7행에서 p=x;에 의하여 p에 a[0]의 주소가 값주기된다. 지적자변수의 앞에 *가 붙으면 그 지적자변수의 내용을 의미한다.

i가 0일 때 10행에서 s1=s1+*(p); 은 s1변수에 p지적자가 가리키는 주소의 내용을 더하라는 뜻이다. 즉 a[0]의 내용이 더해진다.

i가 2일 때 10행에서 s1=s1+*(p+2);에 의하여 a[2]의 내용이 s1에 더해진다. 마찬가지로 i가 4일 때 s1=s1+*(p+4)로 되며 a[4]의 내용이 s1에 더해진다.

i가 0일 때 11행의 s2=s2+*(p+1);에 의하여 a[1]의 내용이 s2에 더해진다.

마찬가지로 i가 2일 때 11행의 s2=s2+*(p+3);에 의하여 a[3]의 내용이 s2에 더해지며 i가 4일 때 s2=s2+*(p+5);에 의하여 a[5]의 내용이 s2에 더해진다.

배렬과 지적자의 리용에서 주의할 점은 다음과 같다.

배렬이름을 지적자로 리용할 때 배렬이름을 증가 또는 감소연산자를 가지고 연산하면 오유를 발생시킨다. 즉 우의 례에서 a++, a--로 쓸수 없다.

그것은 배렬이름 a는 배렬의 선두주소이므로 a++연산이 진행되면 a[0]이 들어있는 장소가 달라지기때문이다. a[0]의 주소가 달라지면 a[0]이 어디에 들어있는가를 알수 없게 된다.

3. 문자렬과 지적자

문자렬은 문자들의 1차원배렬이다. 문자형배렬은 곧 문자렬로 된다.

배렬이름은 배렬선두주소를 나타내는 지적자변수와 같다. 그러므로 문자렬변수는 문자렬지적자변수로 선언하든가 또는 문자형배렬변수로 선언하면 된다.

문자렬지적자에 의한 문자렬변수의 선언형식:

```
char *변수이름;
```

문자형배렬변수에 의한 문자렬변수의 선언형식:

```
char 변수이름[문자개수+1]
```

문자형배렬변수에 의한 문자렬변수의 선언형식에서 문자개수보다 하나 더 많은 기억구역을 확보하는것은 문자렬의 제일 마지막에 문자렬의 끝을 의미하는 기호 \0이 있어야 하기때문이다.

례 1: 문자렬지적자변수를 리용하여 문자렬을 관리하는 프로그람을 작성하여라.

```
#include <stdio.h>
void main(void)
```

```
  {
      char *pt;
      pt="abcdef";
      for(;*pt!='\0';)
       printf("%c",pt++);
        printf("%s\n",pt);
}
```

례 2: 문자형배렬을 리용하여 문자렬을 관리하는 프로그람을 작성하여라.

```
#include <stdio.h>
void main(void)
{
 char pattern[30];
 int i;
 pattern[0]= 'a';
 pattern[1]= 'b';
 pattern[2]= 'c';
 pattern[3]= 'd';
 pattern[4]= 'e';
 pattern[5]= 'f';
 pattern[6]= 'g';
 pattern[7]= '\0';
 for(i=0;pattern[i]!= '\0';i++)
   printf("%c",pattern[i]);
   printf("\n");
   printf("%s\n",pattern);
     }
```

례 1은 문자렬지적자변수를 리용하여 문자렬을 관리하는 방법이며 례2는 문자형배렬변수를 리용하여 문자렬을 관리하는 방법이다.

례 1에서 pt= "abcdef";라는 명령문은 "abcdef"라는 문자렬을 pt에 값주기하는것이 아니라 문자렬이 들어있는 첫 주소를 pt에 준다는것을 의미한다. 즉 주어진 문자렬을 빈 기억구역에 먼저 복사하고 그 주소를 pt에 값주기하는것이다.

문자렬처리함수에서 문자렬을 쓸 위치에는 모두 문자렬지적자로 바꾸어쓸수 있다. 우리가 이미 앞에서 학습한 문자렬처리함수외에 문자렬검색함수에 대하여 보면서 문자렬지적자에 대하여 구체적으로 보기로 하자.

1) 문자렬에서 문자검색

형식:

```
char *strchr(const char *string, char c);
```

기능:

string문자렬에 c로 지적된 문자가 있는가를 검색하고 있다면 그 위치를 돌려준다. 검색이 실패이면 NULL값을 돌려준다.

례 3: char *c, *d;
  c=strchr("Helloworld", "o");

이 례에서는 Helloworld문자렬에 o가 있는가를 검색하여 결과로 5의 결과를 준다.

2) 문자렬에서 문자렬검색

형식:

```
char *strstr(const char *s1, const char *s2);
```

기능:

s1로 지정된 문자렬안에 s2로 지정한 문자렬이 있는가를 검색하여 그 위치를 돌려준다. 검색자료가 없으면 NULL값을 돌려준다.

례 4: 10명 학생의 이름을 읽고 그것을 자모순으로 배렬하는 프로그람을 작성하여라.

```
#include <stdio.h>
#include<string.h>
#define length 20
#define size 30

void main(viod)
{
  char name[size][length];
  int i, j;
  for(i=0; i<size-1;i++)
  for(j=i+1; j<size;j++)
   if((c=strcmp(name[i], name[j]))>0)
{
   char ws[length];
   strcpy(ws,name[i]);
```

```
      strcpy(name[i], name[j]);
      strcpy(name[j],ws);
     };
   for(i=0;i<size;i++)
      printf("%s\n",name[i]);
   }
```

4. 구조체와 지적자

구조체에도 지적자를 적용할수 있다.

구조체도 하나의 자료형이므로 구조체형지적자에 의해 구조체변수를 조작할수 있다.

구조체형지적자도 선언을 해야 리용할수 있다.

구조체형지적자선언형식:

```
struct 구조체이름 *변수이름;
```

례 1:
```
struct student
  {
    char name[10];
    int  age;
    char sex;
  };


struct student s, *a;
a=&s;
```

구조체형지적자변수에 의한 구조체변수의 요소들은 다음과 같이 호출할수 있다.

```
struct 구조체형지적자이름→요소이름;
```

이것은 (*구조체형지적자변수).구조체요소이름과 같이 동작한다.

례 2: 번호, 이름, 성별, 나이, 성적으로 된 한 학생의 구조체에 대한 지적자를 만들고 성적이 최우등인가를 판정하는 프로그람을 작성하여라.

```
include <stdio.h>
main()
  {

struct student
  {
     short num;
     char name[10];
     char sex;
     short age;
     float mark;
   };

struct student *x;
int i;
scanf("%d%s%s%d%f",x→num, x→name[0], x→.sex, x→age, x→mar
      k);
if(*x→SEX='M' && *X→MARK>=4.5)
  printf("%d%s%s%d%f",x→num, x→name[0], x→sex, x→age, x→ma
        rk);
}
```

련습문제

1. 지적자를 리용하여 10개의 실수자료를 가진 배렬 a의 자료들을 다른 배렬 b에 넣는 프로그람을 작성하여라.
2. 지적자를 리용하여 한개의 영어문장을 가진 변수의 내용을 다른 문자렬지적자변수에 넣고 출력하여라.
3. 5명 학생의 3과목 성적자료를 읽고 학생별, 과목별성적을 출력하는 프로그람을 지적자를 리용하여 작성하여라.
4. 20권의 도서에 대한 책이름, 저자, 출판년도, 출판사명을 읽고 2000년이후에 출판된 책의 이름과 출판년도를 출력하는 프로그람을 지적자를 리용하여 작성하여라.
5. 문장과 단어를 입력하여 문장에서 단어의 개수를 구하는 프로그람을 지적자를 리용하여 작성하여라.





콤 퓨 터 상 식

보도부분에서의 콤퓨터의 응용

휴대용정보말단과 콤퓨터망에 의하여 보도의 실시간화와 24시간화가 추진되고 있다. 보도를 받는 립장에서 생각하면 콤퓨터기술에 의하여 보도는 2개의 축으로 진화되여나가리라고 론의되고있다.

하나는 실시간성과 24시간화의 축이며 또 하나는 개별화나 지역성의 중시이다.

보도의 24시간화의 요구는 특히 금융정보 및 기업정보의 측면에서 강하다. 어떤 보도기관의 홈페지에서는 분야별로 어떤 보도를 우선시하여 표시하겠는가를 설정할수 있게 되여있다. 이것을 보도의 개별화라고 한다.

콤퓨터기술의 발전에 의하여 지역중시의 보도도 가능하게 되였다. 인터네트의 보도봉사가운데는 우편번호를 입력하면 해당지역에 대한 보도나 날씨예보를 표시하는것도 있다. 인터네트리용자의 장래적인 확대에 따라 시간 및 종이측면에서의 제약을 뛰여넘는 매체특성에 많은 보도기관들이 기대를 가지고있다고 한다. 인터네트에서의 보도봉사를 장래의 전자신문을 위한 준비로 보고있는 보도기관도 있다고 한다.

제2장. 함수의 리용

제1절. 함수란 무엇인가

프로그람을 작성할 때 어떤 문제는 오직 하나의 프로그람으로밖에 작성할수 없는가 하면 또 어떤 문제는 쎄타를 뜰 때처럼 여러개의 부분프로그람들로 나누어 작성하고 그 부분프로그람들을《조립》해서 하나의 완성된 프로그람을 만들수 있는것도 있다.

이때 부분프로그람들은 서로 다른 사람들이 맡아 작성해도 되므로 프로그람작성이 쉽고 간단하며 전체 프로그람을 작성하는데 시간이 적게 걸린다. 바로 이것이 여러 부분일감들로 된 프로그람을 작성하는 방법이다. 이때 매개 부분일감들을 하나의 함수로 작성할수 있다.

1. 함수란 무엇인가

수학에서는 수모임을 수모임에로 넘기는것을 함수라고 정의한다.
그러면 C언어에서의 함수란 무엇인가?

례 1: 임의의 수값을 입력하여 그 수의 절대값을 구하는 프로그람을 작성하여라.

```
#include <stdio.h>
main()
{
 float a;
 scanf("%f", a);
  if(a>=0)
       printf("%f",a);
  else
   printf("%f",-a);
}
```

우의 프로그람을 절대값을 주는 표준함수를 리용하여 작성하여보자.

```
#include <stdio.h>
main( )
```

```
{
   float a, b;
   scanf("%f", a);
   b=fabs(a)
   printf("%f", b);
}
```

이 프로그람을 보면 첫번째 프로그람에서 주어진 수가 부가 아닌가 부인가를 판정하는 부분대신 fabs라는 이름을 가진 표준함수를 호출하여 출력하였다.

주어진 수의 절대값을 구하는 프로그람은 체계내에 미리 작성되여있다.

6행에서 식 b=fabs(a)는 수값표준함수를 호출하는 식인데 이때 절대값표준함수는 실수 a를 넘겨받아 그의 절대값을 구하여 호출한 측 즉 변수 b에 그 결과값을 되돌려준다.

이처럼 C언어에서는 자주 리용하는 프로그람단위들은 미리 체계내에 작성하여놓고 리용자들은 이것을 호출하여 리용할수 있게 하고있다.

이런 함수들을 표준함수라고 하였다.

이것은 수학에서 정의한 함수와 비슷하다.

례하면 2차함수 $y=2x^2+5x+3$에 대하여 x=-1에서의 함수값을 다음과 같이 구하였다.

$$f(2)=2\cdot(-1)^2+5\cdot(-1)+3=0$$

그러면 프로그람작성자들은 자기의 프로그람안에 프로그람단위들을 작성하여놓고 필요할 때 호출하여 리용하는 방법으로 프로그람을 작성할수 없겠는가?

C언어체계는 이것을 가능하게 하고있다.

례 2: 두 수의 합을 출력하는 프로그람을 함수를 리용하여 작성하여보자.

```
#include <stdio.h>
float sum(float x, float y)
  {
     float z;
     z=x+y;
     return z;
  }
main( )
 {
   float a, b, c;
   scanf( "%f%f" , a, b);
   c=sum(a, b);
     printf("%f", c);
 }
```

해 설

프로그람을 실행하면 8행의 main()함수로부터 실행된다.

11행의 실행시 a, b값을 각각 10, 20으로 입력하자.

12행은 2행부터 7행까지로 이루어지는 sum이라는 이름을 가진 함수를 호출하는 명령문이다.

이때 괄호안의 변수 a, b를 실파라메터라고 하는데 이 값이 함수에로 넘어간다. 즉 값 10, 20이 함수에로 넘어간다.

함수호출명령문에 의하여 프로그람의 실행은 2행으로 이행된다.

2행을 함수머리부라고 한다.

함수머리부의 괄호안에 있는 변수를 가상파라메터라고 하는데 이 변수 x, y가 실파라메터에 있는 변수 a, b값인 10, 20을 각각 넘겨받는다.

4행은 이 함수안에서 쓰이는 변수 z를 선언하는 명령문이고 5행은 두 수 x, y의 합의 결과인 30을 변수 z에 값주기하는 명령문이다.

6행에 의하여 변수 z의 값을 이 함수를 호출한 측에로 되돌려주고 프로그람실행이 이 함수를 호출한 식 다음행에로 이행된다.

즉 변수 z의 값 30이 12행의 변수 c에 넣어지고 프로그람실행은 13행으로 이행된다. 이때 z값을 되돌림값이라고 한다.

13행에 의하여 결과값 30이 출력되고 프로그람은 정상완료된다.

물론 두 수의 합을 구하는 프로그람과 같이 간단한 프로그람은 함수를 리용하지 않고 작성하여도 되지만 보다 복잡한 프로그람은 프로그람을 그 기능별로 나누어 작성하여놓고 필요할 때마다 호출하여 리용하면 프로그람의 작성이 편리해진다.

부분프로그람들은 여러 사람들이 맡아서 작성하여도 되므로 프로그람작성이 쉽고 간단하며 전체프로그람을 작성하는데 시간이 적게 걸린다.

또한 일단 오유가 있거나 갱신할 필요가 있으면 수정하기도 쉽게 된다.

그리고 프로그람안에서 필요할 때마다 이 함수를 호출하여 리용할수 있다.

이때 만든 개개의 프로그람단위를 함수라고 한다.

즉 함수란 돌림값을 가지는 수속으로서 일정한 처리를 진행하기 위한 한개이상의 명령문을 포함하는 프로그람단위이다.

그러면 함수정의형식과 호출형식, 이때 리용되는 매개 인수들의 구체적인 의미를 우에서 작성한 함수를 고찰하면서 보자.

1) 함수의 정의형식

```
돌림값형 함수이름([가상파라메터렬])
{
    [함수에서 사용할 변수들의 정의];
    명령문렬;
    [return [식]];
}
```

함수는 크게 함수머리부와 함수본체로 이루어져있다.

함수머리부는 돌림값형, 함수이름, 가상파라메터렬로 이루어져있다.

함수정의에서 함수머리부는 반드시 있어야 한다.

— 돌림값형

함수는 호출에 의하여 실행되며 그 결과를 자기를 호출한 측에 되돌려준다.

이때 함수가 되돌려주는 값의 자료형이 돌림값형이다.

례 2의 프로그람에서는 함수의 머리부에서 돌림값형을 float형으로 주었다.

그것은 두 실수 x, y의 합의 결과값인 z의 값이 float형인데 return명령문을 통하여 호출한 측의 변수 c값에 float형의 값을 돌려준다는 의미에서 돌림값형을 float형으로 주었다.

함수를 정의할 때 먼저 이 돌림값의 자료형을 정의한다.

```
float sum(float x, float y)
돌림값형
  {
   …
   return z;
   }

main( )   실수값을 되돌린다.
 {
   …
     c=sum(a, b);
   …
 }
```

함수의 돌림값은 C에서 제공하는 모든 자료형이 가능하고 사용자정의형도 가능하다. 또한 매개 자료에 대한 지적자형변수도 함수값으로 넘길수 있다.

함수에서 아무런 값도 되돌리지 않는 경우에는 돌림값형을 void라고 써주면 된다. 또한 함수의 돌림값형이 생략되면 옹근수형(int)으로 번역된다.

함수머리부에서 준 돌림값형은 return문에 있는 식의 형과 일치하여야 한다.

— 함수이름

함수이름은 하나이상의 함수들을 식별하기 위한 이름으로서 C언어의 이름짓기규칙에 따라 결정할수 있으며 이때 그 함수가 어떤 기능을 수행하는 프로그람인가 하는 의미를 담아서 결정할수 있다.

우의 프로그람은 두 수의 합을 구하는 함수라는 의미에서 sum으로 결정하였다.

```
float sum(float x, float y)
      함수이름
```

함수이름은 변수이름과 마찬가지로 함수가 할당되여있는 기억기의 주소이다.

즉 기억기에 적재된(기억된) 명령문들이 놓여있는 첫 주소를 의미한다.

— 가상파라메터렬

가상파라메터렬에서는 이 함수가 호출될 때 이것을 호출하는 측에서 넘겨주는 실파라메터를 받기 위한 파라메터들을 정의한다.

파라메터라는것은 일반적으로 변수를 의미한다고 생각하면 된다.

례 2의 프로그람에서 보면 가상파라메터렬에 있는 변수 x, y는 호출측의 실파라메터인 실수 a, b값을 각각 넘겨받는다는 의미에서 실수형으로 선언하였다.

```
float sum(float x, float y)
  {
   …
   }
main( )
 {
   …
     c=sum(a, b);
 }
```

각각 실수를 넘겨받는다.

가상파라메터의 개수는 적어도 0개이상이다. 즉 하나도 없을수도 있으며 있을수도 있다. 가상파라메터의 정의는 변수정의와 같으며 여러개인 경우에는 반점(,)으로 구분하여 소괄호안에서 정의할수 있다.

소괄호안에서 가상파라메터를 정의할 때 변수이름만을 정의하고 형선언은 함수이름 다음에 할수 있다.

례 3:
```
float  sum(x, y)
int x, y;
{
…
}
```

실파라메터렬과 가상파라메터렬을 좀더 자세히 고찰하여보자.

실파라메터에 있는 변수의 이름과 가상파라메터렬에 있는 변수의 이름은 달라도 된다. 그러나 개수와 순서, 자료형은 1:1대응되여야 한다.

실파라메터렬과 가상파라메터렬에 있는 변수의 이름은 달라도 되기때문에 이미 작성한 프로그람을 수정할 때 간단히 수정할수 있다.

이것은 함수의 독립성이 강하다는것을 말하여준다.

가상파라메터를 가지지 않는 함수는 가상파라메터렬자리에 ( ) 또는 void라고 쓴다.

— 함수본체

함수본체는 {으로 시작되고 }으로 끝난다.

함수본체에서는 이 함수안에서 사용할 변수를 선언하고 필요한 처리를 진행하는 명령문렬을 서술한다.

우의 프로그람에서는 이 함수안에서 사용하는 변수 z를 실수형으로 선언하고 두 수 x, y의 합의 결과를 변수 z에 값주기하였다. 그리고 return명령문을 통하여 결과값을 자기를 호출한 측에 돌려주고 함수실행을 완료하게 하는 명령문을 서술하였다.

```
float sum(float x, float y)
  {
     float z;// 함수안에서 리용할 변수선언
     z=x+y;// 처리내용 서술
     return z;
   }
```

— return명령문

일반형식:

return [식];

이 명령문은 함수의 실행을 끝내고 이 함수를 호출한 측에로 돌아가게 한다.

우의 프로그람에서는 return z; 명령문에 의하여 결과값 30을 호출한 측 즉 변수 c에 되돌리고 프로그람실행을 13행에로 이행한다.

만일 식이 있으면 그 식을 평가하고 그 값을 돌림값을 통하여 돌려주지만 식이 없으면 아무런 값도 돌려주지 않는다.

그러나 이런 경우에도 돌림값형은 존재한다. 이 경우 돌림값형은 void로 된다.

return문에서 리용되는 식의 결과형과 함수정의에서 서술한 돌림값형은 일치하여야 한다.

2) 함수의 호출

함수는 그자체로서는 아무런 일도 할수 없으며 반드시 자기를 호출하는 식에 의하여서만 실행될수 있다.

C언어에서의 함수의 호출에는 되돌림값을 요구하는 호출형식과 되돌림값을 요구하지 않는 호출형식이 있다.

되돌림값을 요구하는 함수호출의 일반형식:

변수=함수이름([실파라메터렬]);

이 명령문에 의하여 호출된 함수의 결과값이 변수에 넣어지게 된다.

함수의 호출은 함수이름뒤에 ( )를 붙이고 그안에 실파라메터값들을 반점(,)으로 구분하여 써넣으면 된다. 실파라메터가 요구되지 않을 경우에는 ( )만을 쓴다.

함수이름은 함수정의에서 정의한 함수이름과 일치하여야 한다.

실파라메터렬은 함수정의에서 선언한 가상파라메터렬에 넘겨주는 실제적인 값이며 가상파라메터의 개수와 순서, 형들이 일치하여야 한다.

함수호출문은 함수를 호출하는 함수안에서 서술한다.

함수호출명령문에 의하여 함수가 호출되고 실파라메터에 있는 변수들의 값이 가상파라메터의 변수들에로 넘어간다.

우의 프로그람에서는 실파라메터인 변수 a, b의 값이 가상파라메터의 변수 x, y의 값으로 넘어간다.

```
float sum(float x, float y)
             ↑
  {          |
   …         |
   }         |  함수호출
             |
main( )      |
 {           |
   …         |
     c=sum(a, b);
   …
 }
```

되돌림값이 요구되지 않는 함수호출의 일반형식:

```
함수이름([실파라메터렬]);
```

례 4: Let’s learn C를 10번 출력하는 프로그람을 함수를 리용하여 작성하여라.

```
#include <stdio.h>
disp()
{
   printf(“Let’s learn C”);
}
main()
{
int i;
    for(i=1; i<=10; i++)
      disp()
}
```

이 프로그람에서는 함수를 호출할 때 아무런 인수도 넘기지 않고 또 함수본체에서 되돌림값이 없다.

례 5: 조합의 총수를 구하는 프로그람을 작성하여라.

```
#include <stdio.h>
int factorial(int m)
{
    int i, p;
    p=1;
    for(i=1; i<=m; i++)
         p=p*i;
    return p;
}

main( )
{
    int n, r, y;
    scanf("%d%d",n, r);
    y=factorial(n)/(factorial(n-r)*factorial(r));
    printf("%d", y);
}
```

3) 함수의 형선언

함수는 main()함수전에 정의하는것이 일반적이다.
그러나 필요에 따라 함수정의를 main()함수안에서 할수도 있다.
이때에는 main()함수전에 함수의 형선언을 하여야 한다.

함수의 형선언은 함수호출에 필요한 함수이름, 가상파라메터의 자료형, 돌림값형을 알려주어 함수호출시 가상파라메터와 실파라메터들과의 형검사를 진행하기 위한것이다. 따라서 함수의 정의가 함수의 호출보다 앞서는 경우는 함수의 형선언을 하지 않아도 되지만 그렇지 않은 경우는 반드시 함수의 형선언을 해야 한다.

함수의 형선언형식:

돌림값형 함수이름([가상파라메터렬]);

함수의 형선언에서는 가상파라메터렬을 이루는 변수들의 이름은 생략해도 된다.
그러나 형이름만은 정확히 반영해야 한다.
함수의 형선언문의 형태와 함수정의문의 형태는 일치하여야 한다.

례 6: 두 수가운데서 큰 수를 출력하는 프로그람을 함수를 리용하여 작성하여라.

```
#include <stdio.h>
float swap(float, float);
main()
   {
      float a, b, c;
      scanf(“%f%f”, a, b);
      c=swap(a, b);
      printf(“%f”, c);
   }

float swap(float x, float y)
  {
     float t;
     t=(x>y) ? x,y;
     return t;
  }
```



탐 구

함수의 정의보다 함수의 호출이 먼저 진행되는 경우 어떤 현상이 일어나는가?

4) main()함수

main()함수는 C언어로 작성하는 프로그람에서 주함수이다.

main()함수는 { 로 시작되며 } 로 끝난다.

함수의 호출을 비롯한 기본적인 명령문들은 다 이 함수안에서 서술하며 프로그람을 실행하면 이 함수부터 실행이 진행된다.

결국 C프로그람은 main()함수와 개별적인 함수들의 결합 즉 함수들의 모임이라고 말할수 있다.

함수를 정의하고 리용함에 있어서 우에서 강조한 문제외에도 다음의 몇가지에 주의를 돌려야 한다.

① 함수는 임의의 순서로 정의하여도 된다.

례 7: 두 수의 합과 차를 구하는 프로그램을 다음과 같이 작성하자.

```
#include <stdio.h>
float sum(float x, float y)
  {
     float z:
     z=x+y:
     return z:
   }

float def(float x, float y)
  {
     float z:
     z=x-y:
     return z:
   }

main( )
  {
     float a, b, c, d;
     scanf("%f%f",a, b);
     c=sum(a, b);
     printf("%f", c);
     d=def(a, b);
     printf("%f", d);
   }
```

이 프로그램은 다음과 같이 작성하여도 된다.

```
#include <stdio.h>
float def(float x, float y)
  {
     float z:
     z=x-y:
     return z:
   }
```

```
float sum(float x, float y)
  {
      float z:
      z=x+y:
      return z:
   }

main()
  {
      float a, b, c, d;
      scanf("%f%f",a, b);
      c=sum(a, b);
      printf("%f", c);
      d=def(a, b);
      printf("%f", d);
   }
```

② 함수의 호출이 정의보다 앞서지 않도록 하며 그렇지 못한 경우에는 철저히 함수의 형선언을 하여야 한다.

③ 함수안에서 또 다른 함수를 정의할수 없다.

우의 프로그람을 다음과 같이 작성하면 오유로 된다.

```
#include <stdio.h>
 float def(float x, float y)
  {
      float z:
      z=x-y:
      return z:
      float  sum(float x, float y)
         {
            float z:
            z=x+y:
            return z:
         }
      }

  main()
  {
      float a, b, c, d;
      scanf("%f%f",a, b);
      c=sum(a, b);
```

```
        printf(“%f”, c);
        d=def(a, b);
        printf(“%f”, d);
    }
```

우의 프로그람은 두 수의 차를 구하는 함수안에서 두 수의 합을 구하는 함수를 정의하였는데 이것은 오유로 된다.

④ 한개의 함수안에서 또 다른 함수를 호출할수 있다.

례 8: $\sin x=1-x^3/3!+x^5/5!+\cdots+x^{(2n-1)}/(2n-1)!$을 마디의 값이 0.001보다 작을 때까지 구하는 프로그람을 작성하여라.

```
#include <stdio.h>
int factorial(int m)
   {
        int i, p:
        for(i=1;i<=m;i++):
           p=p*i
        return p:
      }

 float power(float x,  int  n)
   {
        return x^n :
      }

main()
   {
        float  x, s=1, n, iterm;
        scanf(“%”, x);
        while(iterm>=0.001)
          {
            n=n+1;
            iterm=(-1)^n*pow(2n-1)/factorial(2n-1);
             s=s+iterm
          }
        printf(“%f”, s);
    }
```

⑤ 또한 한개의 함수에서 자기자체를 호출할수도 있다.

2. 함수에서 파라메터넘기기

함수에서의 파라메터의 넘기기는 함수에서 중요한 역할을 한다.

그러므로 이에 대한 옳바른 리해를 가지는것은 함수의 작용을 보다 정확하게 파악하고 활용할수 있게 하며 앞으로 배우게 되는 지식을 원만히 소유하고 활용할수 있게 한다.

파라메터는 호출하는 함수와 호출되는 함수사이에 오고가는 자료들을 전달하는 기능을 수행한다.

파라메터가 선언되는 장소(호출하는 함수에 선언되였는가, 호출되는 함수에 선언되였는가)에 따라 실파라메터와 가상파라메터로 나눈다.

실파라메터는 호출하는 함수에서 호출받는 함수에로 넘기는 변수를 말한다.

가상파라메터는 호출받는 함수에서 호출하는 함수로부터 넘겨받는 값을 보관하는 변수를 말한다.

파라메터자료의 특성에 따라 값에 의한 넘기기와 주소에 의한 넘기기로 분류할수 있다.

1) 값에 의한 넘기기

C언어에서의 표준적인 파라메터넘기기방식은 값에 의한 넘기기(Call by value)이다. 값에 의한 파라메터주고받기는 실파라메터의 실제적인 값들이 대응하는 가상파라메터로 넘어가는 방식이다.

지금까지 우리가 학습한 함수는 값에 의한 넘기기이다.

레 1: 다음의 프로그람의 결과를 고찰하여보자.

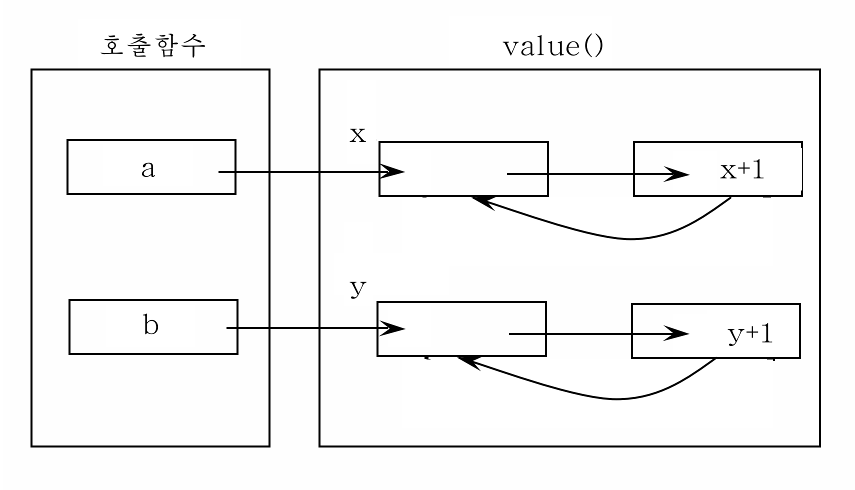
```
#include <stdio.h>
value(float x, float y)
 {
    x=x+1;
     y=y+1;
 }
main( )
 {
   float  a=10. b=20;
   value(a, b);
   printf( “%f%f” a, b);
  }
실행결과 10, 20
```

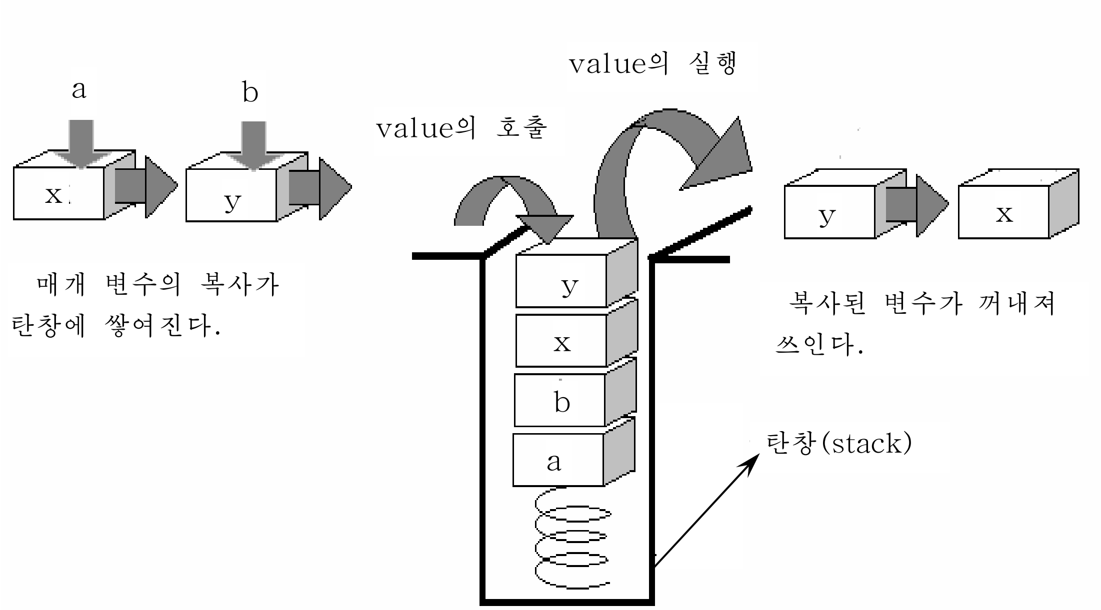
우의 프로그람에서 보면 함수 value안에서 가상파라메터의 값은 4, 5행에 의하여 각각 11, 21로 되였다.

그런데 함수호출측인 main()함수에서 11행의 실행으로 실파라메터 a, b의 값을 출력하여보면 그 값은 각각 10, 20으로 출력된다.

이것은 가상파라메터의 변화가 실파라메터에 영향을 주지 않는다는것을 의미한다.

파라메터넘기기 과정을 그림으로 보면 다음과 같다.





그림에서 보다싶이 함수 value의 호출시에 실파라메터 a, b는 자기의 값을 가상파라메터 x, y에 줄뿐 자기자체에서는 아무런 움직임도 없다.

실파라메터 a, b의 값은 가상파라메터 x, y에 복사되여 탄창령역에 쌓여진다.

함수 value의 실행시에는 가상파라메터의 값이 탄창으로부터 꺼내져 연산이 진행된다. 결국 함수 value의 실행은 실파라메터의 변화에는 아무런 영향을 주지 못한다.

값에 의한 파라메터주고받기에서는 실파라메터의 값이 호출되는 함수의 가상파라메터에 그대로 복사된다.

때문에 호출된 함수는 호출한 함수에 선언된 실파라메터의 기억주소를 알지 못하므로 호출된 측에서는 가상파라메터의 값을 변경할수 있지만 호출한 함수의 실파라메터의 값을 그대로 유지한다.

즉 값에 의한 파라메터주고받기에서는 호출하는 함수의 실파라메터값이 호출된 측에 의하여 변화될수 없다는것이다.

그러므로 값에 의한 파라메터주고받기에서는 가상파라메터를 변경시켜도 실파라메터의 값은 그대로 유지된다. 값에 의한 파라메터넘기기는 자료보호를 목적으로 할 때 리용하며 하나의 값을 되돌릴 때 리용하면 좋다.

값에 의한 넘기기에서 파라메터들의 넘기기과정은 다음의 특징을 가진다.

ㄱ) 가상파라메터 x, y를 위한 림시기억장소가 할당된다.

ㄴ) 실파라메터 a, b의 값만이 가상파라메터에 전달된다.

ㄷ) 실파라메터 a, b의 값은 변화되지 않는다.

함수안에서 가상파라메터에 대한 조작은 그 함수안에서만 유효하며 대응하는 실파라메터에는 영향을 주지 않는다. 즉 그 함수를 호출한 함수에는 영향을 주지 않는다.

2) 주소에 의한 넘기기

주소에 의한 파라메터주고받기는 호출하는 함수에 있는 실파라메터의 주소값이 호출되는 함수의 가상파라메터에 복사하는 방법으로 값을 넘겨주고 받기를 하는것을 말한다.

우의 프로그람을 다음과 같이 고쳐 실행시켜보자.

```
#include <stdio.h>
value(float &x, float &y)
 {
   x=x+1;
   y=y+1;
 }
 main( )
  {
    float  a=10. b=20;
    value(*a, *b);
   printf( “%f%f” a, b);
   }
```

실행결과 11, 21

프로그람실행결과는 가상파라메터의 변화가 실파라메터에 영향을 주었다는것을 보여준다.

10행의 함수호출식을 보면 값에 의한 넘기기때와는 달리 실파라메터를 지적자변수로 설정하였다.

이것은 가상파라메터에 변수 a, b의 값을 넘기는것이 아니라 이 변수들이 기억되여있는 주소를 넘기기 위한 조작을 의미한다. 또한 가상파라메터렬의 변수 x, y앞에도 주소변수표시 &를 붙여 변수를 선언하였다.

이것은 실파라메터의 변수들의 기억주소를 받기 위한 조작이다.

이렇게 하면 함수호출시 실파라메터의 값이 가상파라메터로 넘어가는것이 아니라 실파라메터의 기억주소가 가상파라메터의 변수로 넘어가게 된다. 그러므로 호출되는 함수에서는 호출하는 함수의 실파라메터주소를 알고있으므로 실파라메터의 값을 직접 변경할수 있다.

다시말하여 호출되는 함수에서 가상파라메터의 값을 변경시키면 실제적으로 실파라메터의 값을 변경시킨다.

결론은 주소에 의한 파라메터주고받기에서는 호출하는 함수의 실파라메터주소를 직접 조작하므로 호출되는 함수에서 가상파라메터의 값을 변경시키고 호출하는 함수로 돌아가면 실파라메터의 값도 변화된다는것이다.

주소에 의한 파라메터주고받기의 우점

① 돌림값을 하나이상의 호출하는 함수에로 넘겨줄수 있다는것이다.

일반적으로 값에 의한 파라메터넘기기에서는 return명령문에 의하여 하나의 돌림값을 넘겨준다. 돌림값이 2개이상일 때에는 지적자변수로 된 파라메터들을 선언하고 그것을 통하여 넘겨줄수 있다.

② 또 하나의 우점은 호출하는 함수와 호출되는 함수사이에 기억기를 공동으로 리용하므로 기억기리용률을 높일수 있다는것이다.

결함은 주소를 관리하므로 호출하는 함수의 자료를 변화시킬 위험이 있으므로 예상치 않은 오유를 범할수 있다는것이다.

값에 의한 넘기기와 주소에 의한 넘기기의 차이점

값에 의한 넘기기는 가상파라메터의 내용이 변하여도 실파라메터에는 영향을 주지 않으며 돌림값을 가지는 경우 하나의 돌림값을 가진다.

또한 실파라메터와 가상파라메터가 각각 기억공간을 차지하게 되므로 기억용량손실을 보게 된다.

주소에 의한 넘기기에서는 실파라메터에 있는 변수의 주소를 넘기기때문에 가상파라메터의 변화는 실파라메터에 영향을 주며 하나이상의 돌림값을 가질수 있다.

례 2: 두 자연수의 최대공통약수와 최소공통배수를 구하는 프로그람을 함수를 리용하여 작성하여라.(함수안에서는 유클리드련제법으로 두 수의 최소공통약수를 구하는것으로 하여라.)

```
1: #include <stdio.h>
2: int abc(int &x, int &y)
3: {
4:     if(x<y)
5:     {
6:         int z, r;
7:         z=x:
8:         x=y;
9:         y=z;
10:        }
11:    r=x;
12:    while(r<>0)
13:            {
14:                r=x mod y;
15:                x=y;
16:            }
17:     }
18:     main()
19:    {
20:        int a, b, c;
21:        scanf("%d%d",a, b)
22:        c=a*b;
23:        printf("%", abc(*a, *b));
24:        printf("%", c/abc(a, b));
25:     }
```

해 설

우의 프로그람은 유클리드련제법으로 두 수의 최대공통약수를 구하는 함수를 작성하고 그것을 호출하는 방법으로 작성하였다.

이때 2행에서 보는것처럼 함수호출은 주소넘기기에 의하여 진행하였기때문에 가상파라메터의 변수들의 값의 변화는 실파라메터의 변수에 영향을 준다.

그러므로 22행에서 처음 입력한 두 수의 적을 c라는 변수에 넣어두었다가 24행 즉 두 수의 최소공통배수를 구할 때 리용하였다.

련습문제

1. 물음에 대하여 알맞는 항목을 선택하여라.

1) 함수 abc가 다음과 같이 정의되였다.

```
void abc()
{......}
```

함수정의에서 void의 의미는 무엇인가?

○ 함수 abc를 실행한 후 함수는 돌림값이 없다.

○ 함수 abc를 실행한 후 함수는 되돌아가지 않는다.

○ 함수 abc를 실행한 후 임의의 자료형을 돌릴수 있다.

2) int p(); 일 때 p는 ( )

○ int형변수이다.

○ 함수 p의 호출이다.

○ 함수의 선언, 이 함수의 돌림값은 int형의 값이다.

○ 강제자료형변환문의 변수이다.

3) C언어에서 함수는 ( )

○ 함수안에서 정의가능하다.

○ 함수안에서 호출할수 없다.

○ 함수안에서 호출할수 있지만 자기는 호출할수 없다.

○ 함수안에서 호출할수 있으며 자기도 호출할수 있다.

4) C언어에서 함수의 돌림값의 자료형은 ( )으로 규정된다.

○ return문에 있는 명령문의 자료형

○ 이 함수를 호출한 주함수의 자료형

○ 함수를 호출할 때 림시적

○ 함수를 정의할 때 지정한 함수자료형

5) 아래에서 정확한 함수형식은 어느것인가?

○ double fun(int x, int y)
```
{z=x+y; return z;}
```

○ fun(int x,y)
```
{int z;
return z;}
```

○ fun(x,y)
```
{int x,y; double z;
z=x+y; return z;}
```

○ double fun(int x, int y)

```
   {double z;
   z=x+y; return z;}
```

6) 아래의 말에서 정확한것은 어느것인가?

- ○ 함수를 정의할 때 형식파라메터의 자료형선언을 함수본체내에서 할수 있다.
- ○ return뒤의 값은 명령문이 될수 없다.
- ○ 만일 함수값의 자료형과 돌림값명령문의 자료형이 서로 다르면 함수값의 자료형을 기준으로 한다.
- ○ 만일 형식파라메터와 실파라메터의 자료형이 일치하지 않으면 실파라메터의 자료형을 기준으로 한다.

7) C언어에서는 함수값의 자료형의 정의를 생략할수 있다. 이때 이 함수값의 암시적인 자료형은 ( )

- ○ float형이다.
- ○ int형이다.
- ○ long형이다.
- ○ double형이다.

8) 아래의 함수호출문에서 실파라메터의 개수는 몇개인가?

```
func((exp1,exp2),(exp3,exp4,exp5));
```

- ○ 1
- ○ 2
- ○ 4
- ○ 5

9) 아래의 프로그람의 기능은 함수 F(x,y,z)=(x+y)/(x-y)+(z+y)/(z-y)의 값을 계산한다. 빈칸에 채울 내용을 선택하여라.

```
#include <stdio.h>
#include <math.h>
float f(float, float);
void main()
{
float x, y, z, sum;
scanf("%f%f%f", &x, &y, &z);
sum=f(①) + f(②);
printf("sum = %f\n", sum);
}
float f(float a, float b)
{
```

```
float value;
value = a / b;
return (value);
}
```

①

- x-y,x+y
- x+y,x-y
- z+y,z-y
- z-y,z+y

②

- x-y,x+y
- x+y,x-y
- z+y,z-y
- z-y,z+y

2. 제시된 문장을 완성하여라.

아래의 add함수의 기능은 두 파라메터의 합을 구하고 그 값을 호출한 함수에 돌려준다. 함수에서 틀린부분은 ( ), 그것을 고치면 ( )

```
void add(float a, b)
    {
        float c;
        c = a + b;
        return c;
    }
```

3. n^p을 계산하는 프로그램을 작성하여라. 수 n의 p제곱을 계산하는 함수 power()를 정의하여라.
4. 시간을 초로 변환하는 hms_to_secs()함수를 정의하여라.
5. 함수가 호출되면 그것이 몇번 호출되었는가를 표시하는 다음과 같은 형식의 통보문을 화면에 출력하는 함수를 정의하여라.

"3번 호출하였습니다."

그다음 적어도 이 함수를 10번 호출하는 프로그램을 작성하여라.





콤 퓨 터 상 식

콤퓨터프로그람은 어떻게 만들어지는가

프로그람이란 콤퓨터가 수행하여야 할 명령들을 서술해놓은것으로서 다음과 같은 단계를 통하여 작성되여야만 요구하는 처리결과를 정확히 얻을수 있게 된다. 첫 단계는 무엇을 처리할것인가를 정확히 파악하는 단계이다. 이 단계를 문제의 정의단계라고도 하는데 무엇을 할것인가를 정확히 정의해야 목적하는 결과를 얻을수 있다. 둘째 단계는 어떤 자료가 입력되고 출력되여야 하는가를 정의하는 단계이다. 이 단계를 입출력설계(Input/Output Design)라고도 하는데 첫 단계에서 정의된 문제를 해결하기 위하여 어떤 자료가 입력되여야 하며 또한 결과로 어떤 자료들이 출력되여야 하는가를 결정하여야 한다. 셋째 단계는 자료를 어떻게 처리할것인가에 대하여 설계하는 단계이다. 처리설계(Process Design) 또는 프로그람설계단계라고도 하는데 기본은 둘째 단계에서 정해진 입력자료들을 어떤 과정으로 처리해야 하는가에 대한 알고리듬설계이다. 이 단계에서는 프로그람의 효률적인 작성과 처리과정에 대한 명확한 분석을 위하여 흐름도(Flow Chart)와 의사코드(Pseudo Code)방식이 사용된다. 넷째 단계는 원천프로그람을 작성하는 단계이다. 이 단계를 코드화(Coding)라고 한다. 원천프로그람이란 프로그람작성언어로 작성된 프로그람을 의미한다. 원천프로그람을 작성한 다음에는 그것을 기계어로 번역하고 실행파일로 만들어 프로그람의 정확성에 대한 검사단계를 거쳐 완성한다.

제2절. 변수의 적용범위

범위(Scope)란 《포괄하는 한계의 테두리 또는 그 내부》라는 뜻으로서 일반적으로는 《구역》, 《령역》 등을 의미하는 말로 쓰인다.

《프로그람작성에서 나는 학교적으로 단연 1등이야!》라는 말에서 《학교적으로》라는 말은 《학교범위》라는 뜻으로서 범위가 학교안이다.

또한 《이번 시험에서 영철이는 학급적으로 1등을 했어!》라는 말에서는 영철이가 1등을 한 범위가 《학급범위》로 정해지고있다.

이렇게 일상생활에서 우리가 쓰는《학년적으로》,《학급적으로》,《학교적으로》,《구역적으로》, 《전국적으로》 등의 말은 다 범위의 의미를 포함하고있다.

프로그람에서 변수는 중요한 역할을 한다.

프로그람에서 함수는 변수와 함께 일을 한다.

변수는 프로그람작성자에 의하여 선언된 그 시각부터 콤퓨터의 기억기에서 자기의 구역을 할당받고 맡은 일을 수행하게 된다.

그런데 변수가《어디에서》,《얼마동안》,《어떤값을 가지고》 일을 하겠는가 하는 문제는 C프로그람작성에서 매우 중요한 자리를 차지한다.

— 국부변수

레 1: Let's learn C를 출력하는 함수를 만들고 이 함수를 main()함수에서 몇번 호출하였는가를 출력하는 프로그램을 작성하여라.

```
include<stdio.h>
disp()
 {
   int k;
    k=k+1;
      printf("Let's learn C");
 }
 main()
  {
    int  i, k;
    for(i=1; i<=10;i++)
      disp();
      printf("%d", k);
   }
```

프로그램을 실행시키면 결과값이 10이 아니라 임의의 값이 출력된다.

그것은 4행에서 선언된 변수 k는 함수 disp안에서만 적용가능한 변수이고 10행에서 선언된 변수 k는 main()함수안에서만 적용가능한 변수이기때문이며 결국 변수이름은 같아도 서로 다른변수로 된다.

이렇게 어떤 블로크나 함수안에서 선언된 변수를 **국부변수**(local variable)라고 한다.

국부변수는 그것이 선언된 함수나 블로크안에서만 적용가능한 변수이다.

국부변수는 함수나 블로크의 시작에서 생성되고 그 함수나 블로크의 사명이 끝나면 소멸된다.

즉 함수나 블로크안에서 선언된 다음 어떤값을 받아 해당한 처리를 진행한 다음 함수나 블로크의 실행이 완료되면 이 변수에는 원래 주어진 값이 보관되여있는것이 아니라 예측할수 없는 임의의 값이 들어가게 된다.

국부변수 선언형식:

자료형 변수이름

변수가 여러개일 때는 반점으로 구분하여 써줄수 있다. 또한 변수선언시 변수초기화 즉 변수에 값을 넣을수도 있다.

레 2: int a=10;

— 대역변수

우의 프로그람을 다음과 같이 고쳐보자.

```
include<stdio.h>
int k;
disp()
 {
    k=k+1;
    printf("Let's learn C");
 }
main()
{
   int  i;
   for(i=1; i<=10;i++)
     disp();
     printf("%d", k);
 }
```

프로그람을 실행시키면 k의 값이 10으로 정확한 값이 출력된다.

2행에서와 같이 함수안이 아닌 프로그람의 임의의 위치에서 선언된 변수를 대역변수(global variable)라고 한다.

대역변수는 그 프로그람의 모든 함수와 블로크에서 다 적용가능한 변수이다.

이 변수는 프로그람의 실행과 함께 생성되며 그 프로그람의 사멸과 함께 소멸된다. 즉 프로그람작성자에 의하여 선언된 다음 어떤 값을 받아 해당한 처리를 진행한 다음 이 프로그람의 실행이 완료되면 이 변수가 소멸되는데 이때 국부변수와는 달리 수값변수이면 0이, 문자형변수이면 빈문자 " "가 들어간다.

이와 같이 C언어에서 변수는 선언위치에 따라 국부변수와 대역변수로 나눌수 있다.

변수의 적용범위는 선언위치뿐아니라 선언형식에 따라 규정되며 선언형식은 기억기에서 변수가 놓이는 위치와 존재기간에 따라 규정된다.

— 정적변수

우의 프로그람을 다음과 같이 고쳐보자.

```
include<stdio.h>
disp()
 {
   static k;
    k=k+1;
    printf("Let's learn C");
    printf("%d", k);
```

```
   }
  main()
   {
     int  i;
     for(i=1; i<=10;i++)
     disp();
    }
```

이 프로그람을 실행시키면 변수 k의 값이 1, 2, 3, … , 10과 같이 출력된다.

4행에서 선언한 변수 k는 함수안에서 선언한 변수인데 어떻게 매번 변수값이 소멸되지 않고 자기의 값을 보관하고있는가?

이것은 변수 k를 정적변수(static)로 선언하였기때문이다.

정적변수는 국부변수이지만 그 함수의 사명이 끝나도 그 값이 소멸되지 않고 본래의 값을 유지한다.

《정적》이라는 말은 《움직이지 않는다》 또는 《변화발전이 없는》이라는 뜻이다.

정적변수란 국부변수이면서도 언제나 값을 유지할수 있도록 프로그람실행 전기간 존재하는 변수이다.

정적변수 선언형식:

static 자료형 변수이름;

그러면 우의 프로그람에서 변수 k를 대역변수로 선언하면 되겠는데 왜 정적변수를 리용하는가?

그것은 대역변수가 많으면 프로그람이 복잡하여지고 그로부터 오유수정도 어려워지지만 우와 같이 정적변수를 리용하면 프로그람작성이 간편해지고 오유수정도 훨씬 쉬워지기때문이다.

이렇게 변수는 선언형식에 따라 그 적용범위가 달라진다.

— 자동변수

어떤 함수(또는 블로크)안에서 선언되여 유효범위가 그 함수(또는 블로크)로 제한되며 함수의 사명이 끝나면 변수가 자동적으로 소멸되는 변수를 자동변수라고 하며 그 예약어는 auto이다.

예약어는 선언되는 변수의 기억기위치를 가리키는 단어로서 자동변수에 대해서는 예약어를 생략할수 있다. 그러므로 예약어가 없는 국부변수를 자동변수라고 하며 자동변수는 프로그람에서 많이 쓰이는 변수이다.

자동변수는 기억기에서 일시적인 령역을 차지한다.

자동변수 선언형식:

auto 자료형 변수이름;

례 3: auto int k;   // 옹근수형변수 k를 자동변수로 선언
auto float k; // 고정소수점형변수 k를 자동변수로 선언

— 확장변수

확장변수란 함수(또는 블로크)밖에서 선언되여도 리용할수 있는 변수 또는 다른 파일에서도 리용할수 있는 함수나 변수를 말한다.

확장변수는 프로그람번역시 한번 기억기에 할당되여 프로그람전체에 영향을 미치는 변수로서 함수밖에서 변수를 선언할 때 특별히 기억위치에 대한 지적(예약어)이 없으면 확장변수로 선언된다.

그러므로 같은 파일안에서는 대역변수로 볼수 있고 두개이상의 파일들사이에서는 같은 기억장소를 가짐으로써 파일이 서로 련결되여있음을 의미한다.

확장변수 선언형식:

extern 자료형 변수이름;

례 4: extern int i;   // 옹근수형변수 i를 확장변수로 선언
extern void other(); // other()함수를 확장변수로 선언

— 등록기변수

등록기변수란 값의 보존장소가 기억기가 아니고 중앙처리장치(CPU)안의 등록기인 변수이다.

등록기변수 선언형식:

register 자료형 변수이름;

이 변수를 리용하면 CPU의 등록기를 사용하므로 처리속도가 대단히 빠르다.

그러므로 반복처리 등에서 값을 자주 변경해야 할 변수들에 대하여 효과적이다.

등록기변수의 형은 int, char, 지적자형 등이며 변수의 생성과 소멸은 국부변수에서와 같다.

물론 콤퓨터의 기종에 따라 등록기의 개수가 제한되여있으므로 체계에서 쓰지 않는 등록기의 개수를 초과할만큼의 많은 등록기변수를 선언한 경우에 초과된 변수는 자동변수로 기억기에 할당된다.

이와 같이 변수는 선언된 위치와 형식에 따라 그 적용범위가 달라지게 된다.

변수를 리용함에 있어서 다음과 같은 점들을 주의하여야 한다.

- 여러 준위의 변수사용

함수준위의 국부변수는 그 함수안의 블로크에서 사용할수 있다.

즉 함수준위의 국부변수는 그 함수안의 블로크에 관해서는 대역변수로 된다.

- 같은 이름의 변수사용

준위가 서로 다른 경우에 같은 이름의 변수를 얼마든지 사용할수 있으며 이 경우 준위가 좁은것일수록 우선순위가 더 높다.

변수들의 특징

특징 \ 변수분류	자동 (auto)	등록기 (register)	정적(static)		확장 (extern)
			내부	외부	
범 위	국 부	국 부	국 부	대 역	대 역
선언된 블로크에서 실행후의 값	×	×	○	○	○
같은 파일의 다른 함수에서 참조	×	×	×	○	○
다른 파일에서 참조	×	×	×	×	○
초기화하지 않은 결과	임의의 값	임의의 값	0	0	0
초기화회수	실행시 수시로	실행시 수시로	1 회	1 회	1 회
기억장소	탄 창	등록기	기억기	기억기	기억기

련습문제

1. 물음에 대하여 알맞는 항목을 선택하여라.
   1) 하나의 C프로그람원천파일에서 이 파일의 모든 함수에서만 사용되는 변수를 정의하려면 이 변수가 요구하는 기억류형은 무엇인가?
      - ○ extern
      - ○ register
      - ○ auto
      - ○ static
   2) 하나의 함수를 static로 선언한 후에 이 함수는 ( )
      - ○ 같은 원천파일의 함수에서도 호출가능하며 다른 원천파일의 함수에서도 호출가능하다.
      - ○ 같은 원천파일의 함수에서만 호출가능하며 다른 원천파일의 함수에서는 호출불가능하다.
      - ○ 다른 원천파일의 함수에서만 호출가능하며 같은 원천파일의 함수에서는 호출불가능하다.
      - ○ 같은 원천파일의 함수에서도 호출불가능하며 다른 원천파일의 함수에서도 호출불가능하다.
   3) 만일 함수의 블로크안에서 변수를 정의하면 이 변수는 ( )
      - ○ 이 복합문에서만 유효하다.
      - ○ 이 함수에서 유효하다.

○ 이 프로그람범위내에서 모두 유효하다.
○ 비법적인 변수로 된다.

4) 아래에서 정확하지 않은 설명은 어느것인가?
○ 서로 다른 함수에서 같은 이름의 변수를 사용할수 있다.
○ 형식파라메터는 국부변수이다.
○ 함수내에서 정의된 변수는 이 함수범위내에서만 유효하다.
○ 함수의 복합문에서 정의된 변수는 이 함수범위내에서 유효하다.

5) 아래의 프로그람의 정확한 실행결과는 어느것인가?

```
#define max 10
int a[max],i;
void sub2()
{
   int a[max], i, max;
   max = 5;
   for(i = 0; i < max; i++) a[i] = i;
}
void sub1()
{
for(i = 0; i < max; i++) a[i] = i;
}
void sub3()
{
   int i;
   for(i = 0; i < max; i ++) printf("%d ",a[i]);
   printf("\n");
}
void main()
{
   printf("\n"); sub1(); sub3(); sub2(); sub3();
}
```

○ 0 2 4 6 8 10 12 14 16 18
0 1 2 3 4
○ 0 1 2 3 4
0 2 4 6 8 10 12 14 16 18
○ 0 1 2 3 4 5 6 7 8 9
0 1 2 3 4
○ 0 2 4 6 8 10 12 14 16 18
0 2 4 6 8 10 12 14 16 18

6) 함수에서 기억류형을 지정하지 않은 국부변수의 암시적인 기억류형은 ( )
○ 자동변수(auto)이다.
○ 정적변수(static)이다.
○ 확장변수(extern)이다.
○ 등록기변수(register)이다.

2. 프로그램의 실행결과에 대하여 설명하여라.

①

```
#include <stdio.h>
void fun(int p)
{
  int a = 10;
  p = a;
  ++p;
}
void main()
{
  int a = 5;
  fun(a);
  printf("%d\n", a);
}
```

②

```
#include <stdio.h>
int abc(int u, int v);
void main()
{
  int a = 4,b = 16, c;
  c = abc(a,b);
  printf("%d\n", c);
}
int abc(int u, int v)
{
  int w;
  while(v)
  {
    w = u % v;
    u = v;
    v = w;
  }
  return u;
}
```

③

```
#include <stdio.h>
void main()
{
  int a,b;
  a = 5; b = 10;
  printf("before swap a = %d,b = %d\n", a, b);
  swap(a,b);
  printf("after swap a = %d,b = %d\n", a, b);
}
```

```
void swap(int x, int y)
{
  int temp;
  temp = x; x = y; y = temp;
  printf("in swap x = %d,y = %d\n", x, y);
}
```

④

```
#include <stdio.h>
int i = 0;
workover(int i)
{
  i = (i % i) * ((i * i) / (2 * i) + 4);
  printf("i = %d\n", i);
  return (i);
}
reset(int i)
{
  i = i <= 2 ? 5:0;
  return (i);
}
void main()
{
  int i = 5;
  reset(i / 2);     printf("i = %d\n",i);
  reset(i = i/2);   printf("i = %d\n", i);
  reset(i / 2);     printf("i = %d\n", i);
  workover(i);      printf("i = %d\n", i);
}
```

⑤

```
#include <stdio.h>
fun(int a, int b)
{
  static int m = 0, i = 2;
  i += m + 1;
  m = i + a + b;
  return (m);
}
void main()
{
  int k=4, m=1, p;
  p = fun(k, m); printf("%d", p);
  p = fun(k, m); printf("%d", p);
}
```

⑥

```c
#include <stdio.h>
f(int a)
{
  int b = 0;
  static int c = 3;
  a = c++,b++;
  return (a);
}
void main()
{
  int a = 2, i, k;
  for(i = 0; i < 2; i++)
  k = f(a++);
  printf("%d\n", k);
}
```

⑦

```c
#include <stdio.h>
int d = 1;
void fun(int p)
{
  int d = 5;
  d += p++;
  printf("%d", d);
}
void main()
{
  int a = 3;
  fun(a);
  d += a++;
  printf("%d", d);
}
```

⑧

```c
#include <stdio.h>
int k = 1;
void fuc(int m)
{
  m += k; k += m;
 {
  char k = 'B';
  printf("%d, ",k-'A');
 }
 printf("%d,%d, ", m, k);
}
```

```
void main()
{
  int i = 4;
  fuc(i);
  printf("%d, %d\n", i, k);
}
```

3. n개의 세자리수가운데서 수의 모든 자리의 수자들이 다 같은 수들을 골라 작아지는 순서로 배렬하는 프로그람을 작성하여라.

4. 홀수들로 (1), (3, 5), (7, 9, 11), …와 같은 묶음렬을 만들었다. 20번째 묶음까지의 매 묶음을 표시하고 매 묶음의 원소들의 합을 구하는 프로그람을 작성하여라.



콤퓨터상식



최신주변장치를 련결하기 위해서 제기되는 작업

수자식사진기와 같은 최신형장치를 련결하기 위해서는 이런 장치들에서 사용하는 결합부방식을 사용가능상태로 설정해주어야 한다. 아무리 장치설명서를 잘 읽고 장치구동프로그람을 설명서대로 설치한다고 해도 기본입출력체계에 해당 장치를 쓸수 있는 결합부를 등록하지 않으면 장치를 콤퓨터에 련결하여 쓸수 없다.

최신주변장치들은 USB나 IEEE1394와 같은 최신결합부방식에 의하여 콤퓨터에 련결하는 경우가 많다. 그러므로 이런 결합부를 쓸수 있게 기본입출력체계에 등록해야 한다. 그러자면 기본입출력체계의 기본차림표에서 USB포구에 대하여 설정을 Disable로부터 Enable로 바꾸어야 한다.

제3절. 재귀호출

처음값을 알고 그 다음값은 첫값으로부터 구할수 있게 되여있는 문제를 풀어야 할 경우에 자주 부딪치게 된다. 이런 문제를 콤퓨터로 풀 때는 재귀함수라는것을 리용하여 진행한다.

례를 들어 n!을 구하는 문제를 보자.

n!은 다음과 같이 구해진다.

$$n!=n*(n-1)*(n-2)*(n-3)*\cdots*2*1$$

이제 n!을 구하는 함수를 f(n)이라고 하면 우의 식은 다음과 같이 표시할수 있다.

$$n!=n*\underbrace{(n-1)*(n-2)*(n-3)*\cdots*2*1}_{f(n-1)}$$

즉 $n!=n*f(n-1)$로 표현된다.

이것은 다시 다음과 같이 표시할수 있다.

$$n!=n*f(n-1)=n*(n-1)*f(n-2)$$

$0!=1$이므로 $f(0)=1$로 한다면 일반식은 $f(1)=1$, $f(n)=n*f(n-1)$

이것을 리용하여 n!을 구하는 프로그람을 다음과 같이 작성하자.

```
#include <stdio.h>
int fact(int m)
 {
  if(m==0)
     return 1;
  else
     return (m*fact(m-1);
 }
 main()
{
   int n=3, f;
   f=fact(n);
     printf("%d factorial %d",n,f);
 }
```

해설

우의 프로그람을 실행하면 먼저 9행의 main()함수로부터 실행된다.

12행에 의하여 2행의 함수 fact가 호출된다.

4행이 만족되지 않으므로 7행으로 실행이 이행된다. 7행을 보면 실파라메터값이 1만큼 감소한 상태에서 또다시 함수 fact를 호출한다. 즉 자기자신을 호출한다. 이때 변수 m의 값 3은 탄창구역에 쌓여지며 함수호출은 fact(2)로 된다.

4행이 만족되지 않으므로 7행으로 실행이 이행된다. 다시 실파라메터의 값이 1만큼 감소한 상태에서 또다시 함수 fact를 호출한다. 이때 변수값 2는 탄창구역에 쌓여지며 함수호출은 fact(1)로 된다.

4행이 만족되지 않으므로 7행으로 실행이 이행된다. 다시 실파라메터의 값이 1만큼 감소한 상태에서 또다시 함수 fact를 호출한다. 이때 변수값 1은 탄창구역에 쌓여지며 함수호출은 fact(0)으로 된다.

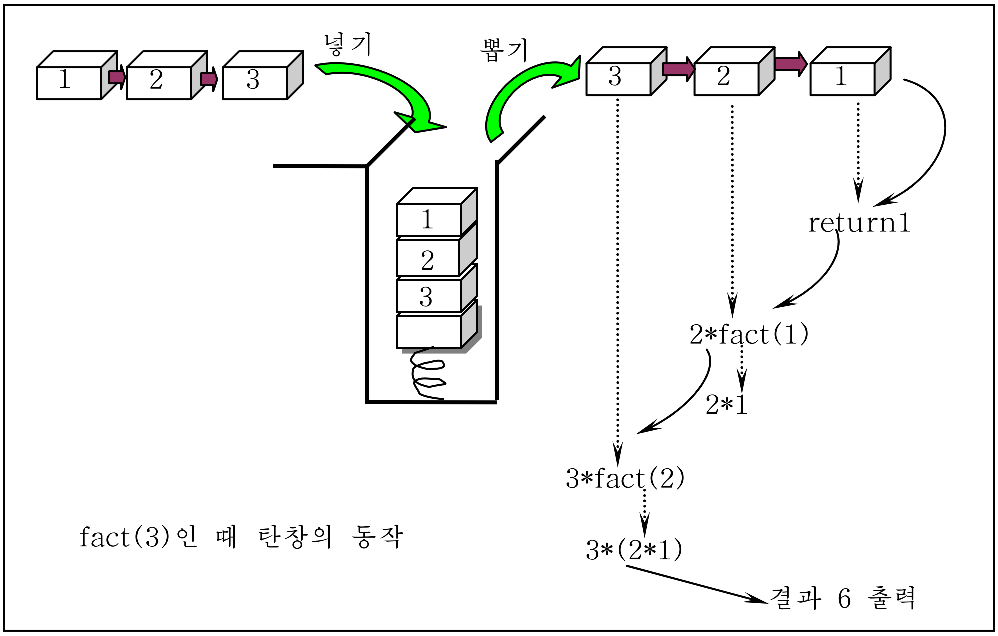
4행이 만족되므로 5행을 실행하며 이것에 의하여 함수에서 빠져나온다. 함수는 자기의 사명이 끝나면 자기를 호출한 측에로 되돌아간다.

fact(0)의 사명이 끝났으므로 return 1; 명령문에 의하여 자기를 호출한 측 fact(1)로 돌아간다. 이때 탄창구역에 쌓여졌던 변수값 1이 꺼내져 자기를 호출한 함수에 값 1을 되돌린다.

fact(1)의 사명이 끝났으므로 return 1*fact(1-1) 명령문에 의하여 자기를 호출한 측 fact(2)로 돌아간다. 이때 탄창구역에 쌓여졌던 변수값 2가 꺼내져 함수값이 2*1로 된다.

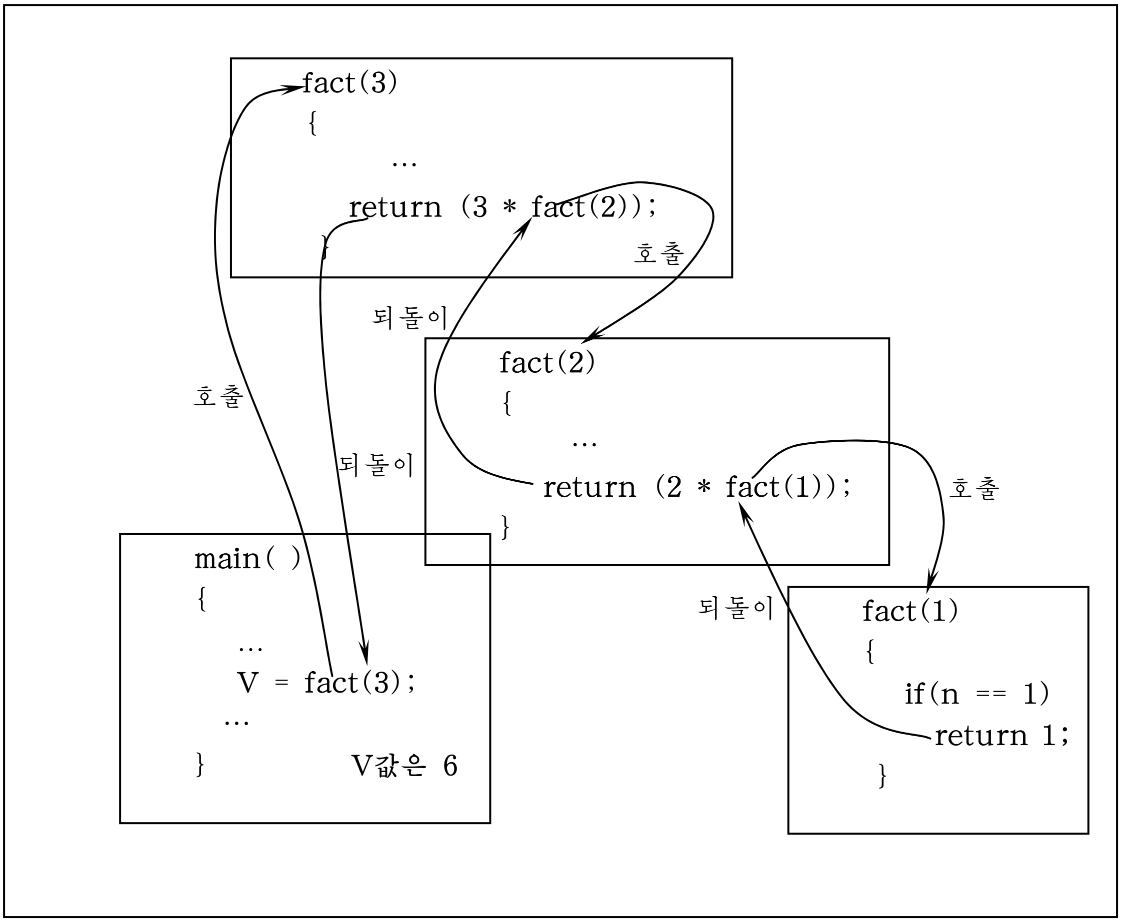
fact(2)의 사명이 끝났으므로 return 2*fact(2-1) 명령문에 의하여 자기를 호출한 측 fact(3)으로 돌아간다. 이때 탄창구역에 쌓여졌던 변수값 3이 꺼내져 함수값이 3*2*1로 된다. 그리하여 12행의 변수 f에 6이 들어간다.

되돌아가는 과정에 변수들의 꺼내기과정을 보면 다음과 같다.



fact(3)인 때 탄창의 동작

함수의 호출과 되돌아오기과정을 종합하여보면 다음과 같다.



이렇게 함수는 자기자신도 호출할수 있다.

이런 함수를 재귀함수라고 한다.

결국 재귀함수란 자기자신을 호출하는 함수를 말한다.

재귀는 함수의 기능인데 프로그람작성에서 널리 리용되는 방법으로 되고있다.

그러면 C프로그람작성에서 재귀가 어떻게 가능한가?

이것은 탄창이라는 기억구역을 리용하여 가능하게 한다.

재귀함수는 자기가 자기자신을 호출하므로 재귀호출이 진행되면 그안에서 쓰이는 변수는 탄창령역에 쌓여진다.

그리고 return문이 실행되면 그때마다 탄창으로부터 변수가 복귀된다.

우와 같이 재귀호출은 제일 처음으로 호출된것이 제일 마지막에 실행되며 제일 마지막에 호출된것이 제일먼저 실행된다.

즉 LIFO(Last In First Out)구조-탄창구조로 실행된다.

탄창이란 실지 총에서의 탄창을 말한다. 탄창에서는 탄창의 제일 우에 있는 탄알

이 제일 마지막에 재워진 탄알이다. 이 탄알은 총을 쏘면 제일먼저 발사된다. 바로 기억기에서의 이러한 실행구조를 탄창구조라고 한다.

재귀호출은 탄창을 사용하므로 재귀호출이 지나치게 많으면 탄창넘침오유가 발생할수 있다.

이것은 프로그람의 효률을 떨어뜨리는 경향이 있다.

따라서 탄창의 크기나 재귀호출의 깊이를 잘 고려하여야 한다.

그러나 프로그람을 간결하고 알기 쉽게 작성하는데서 아주 효과적이며 일반적인 프로그람작성방법으로는 어려운 문제를 손쉽게 작성할수 있는 우점이 있다.

재귀함수인 경우 함수안에는 반드시 재귀호출에서 빠져나오는 방법이 결정되여있어야 한다. 그렇지 않으면 무한히 계속된다.

일반적으로 재귀함수는 점화적인 문제 즉 어떤 초기상태에서 다음단계에로, 다음상태에서 또다른 상태에로 넘어가는 식으로 련속적으로 상태를 변화시켜 최종값에 이르는 문제해결에 리용한다.

례: 주어진 두 자연수의 최대공통약수를 재귀함수를 리용하여 구하는 프로그람을 작성하여라.

두 자연수 a=48, b=36이 주어졌다고 하자.

이 두 자연수의 최대공통약수는 다음과 같이 구할수 있다.

r=a-b=48-36=12

```
#include <stdio.h>
int gcd(int x, int y)
{
if(x==y)
  return x;
else
  {
    if(x>y)
      return  gcd(x-y, y);
    else
      return gcd(y-x, x);
    }
}
```

— 재귀호출방법의 분류

다음의 두 프로그람을 보자.

ㄱ)

```
main()
{
int x = 10;
int value;
value =sum(x);
 …
}

int sum(int n)
{
int temp;
if (n <= 1)
     temp = n;
else
    temp = n + sum(n-1);
return(temp);
}
```

직접재귀

ㄴ)

```
main()
{
    A(…);
}
void A(...)
{
    if (…)
    {
        printf(…);
    }
    else
        B(…);
}
void B(…)
{
    A(…);
}
```

간접재귀

재귀호출방법에서 두 프로그람이 서로 다른 점은:

ㄱ)에서는 함수(sum)가 직접 자기자신을 호출하였다.

ㄴ)에서는 함수(A)가 자기를 호출한 함수(B)를 호출하는 형식으로 자기를 호출하였다. 즉 여기서는 다른 함수를 통하여 자기에 대한 호출이 진행되였다.

ㄱ)형식의 재귀호출을 직접호출, ㄴ)형식의 호출을 간접호출이라고 한다.

재귀함수작성에서 일반적으로 리용되는 방식은 직접호출방식이다.

간접호출은 두개의 함수사이에 이루어지는 재귀로서 많이 쓰이지는 않으나 재귀함수의 구성이 복잡한 경우 부분함수로 구성된 재귀함수를 만들고 재귀를 실현하는데서 효과적으로 리용될수 있다.

련습문제

1. 어떤 수를 화면에 한자리씩 수직으로 내려쓰는 프로그람을 재귀함수를 리용하여 작성하여라.

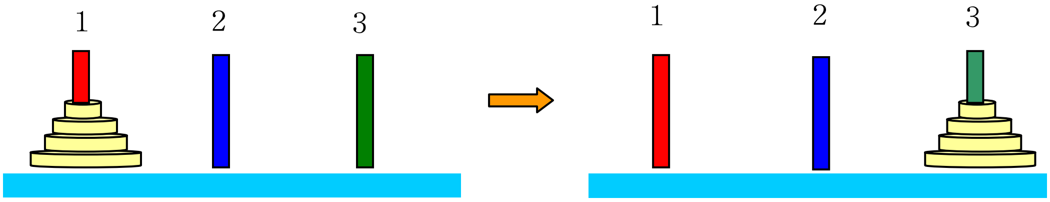
2. 재귀함수를 리용하여 하노이탑문제를 풀어보아라.

   즉 3개의 기둥이 있는데 첫번째 기둥에 n개의 반경이 서로 다른 원반이 끼워져 있으며 큰 원반은 밑에, 작은 원반은 우에 놓여있다. 첫번째 기둥의 원반들을 세번째 기둥으로 옮기는 문제이다.

   옮기기규칙은 다음과 같다.

   옮길 때 두번째 기둥을 리용하여 원반을 림시 끼워넣을수 있다.

   그러나 어느때든지 큰 원반이 밑에 놓이게 하고 작은 원반이 우에 놓이도록 하며 한번에 한개의 원반밖에 옮기지 못한다.







콤퓨터상식

사무프로그람에서 조작이나 화면의 공통화가 실현되고있다

응용프로그람은 문서작성이나 부기계산 등 특정한 업무를 처리하기 위하여 만들어진 프로그람이다. 응용프로그람에는 일반적으로 사용되는 문서작성프로그람, 표계산프로그람, 자료기지프로그람외에 부기계산이나 생활비계산, 창고관리 등 특정한 업무용응용프로그람들이 있으며 그 종류는 다종다양하다.

그가운데서도 최근에 많은 기업소들에서 사용되고있는것이《통합사무프로그람》이다. 이것은 문서작성, 표계산, 자료관리, 발표 등 사무실에서의 업무처리에 많이 리용하는 여러 종류의 응용프로그람들을 한개로 묶은 통합프로그람이다. 개별적으로 매 프로그람들을 구입하는것보다 경제적이며 조작방법이나 화면구성 등도 공통화되여있으므로 일단 습득하면 응용이 쉬운 특징이 있다. 오늘날 기업소나 사무실의 여러대의 콤퓨터들을 국부망으로 련결함으로써 전체적인 생산성을 향상시킬수 있게 되였다. 최근의 통합사무프로그람은 콤퓨터망에 대응되여있어 서로 떨어진 사람들이 콤퓨터상에서 문서를 교환하거나 전자문서로 지령을 주거나 사업보고를 할수 있게 지원하고있다. 통합사무프로그람에서는 한번 작성한 자료를 재리용할수 있으며 보고서용으로 작성한 자료를 발표용으로 리용할수도 있다. 공통적인 조작성을 실현하고 자료의 련계기능 등도 갖춘 통합사무프로그람의 리용가치는 높다.

제3장. 도형그리기

제1절. 자리표계와 색의 조종

1. 영상방식

프로그람으로는 화면에 영어문자와 수자와 같은 기호뿐만아니라 원이나 4각형, 함수의 그라프와 같은 여러가지 도형도 나타낼수 있다.

화면에 기호를 표시하게 한다든가 기호도 표시하고 도형도 그리게 하자면 화면영상방식을 설정해주어야 한다.

프로그람으로 그림을 그리자면 프로그람화면영상방식을 지정해주어야 한다.

화면영상방식이란 화면의 분해능과 그리기를 색방식으로 하겠는가 또는 흑백방식으로 하겠는가 등을 지정하는것이다.

콤퓨터에서 화면에 그림을 표시하자면 내부에 영상표시기판(Adapter)이 반드시 있어야 하는데 기판종류에는 여러가지가 있다.

일반적으로 리용되는 영상표시기판에는 다음과 같은것들이 있다.

CGA(Color Graphics Adapter), EGA(Ehanced Graphics Adapter),

HGC(Herculse Graphics Card), MCGA(Multicolor Graphics Array),

MDPA(Monochrome Display Printer Adapter), VGA(Video Graphics Array)

이외에 CGA, EGA 및 VGA계렬로서 기능이 더높은 OCGA, OEGA, OVGA기판들도 있다.

콤퓨터마다 이가운데서 어느 한개 기판이 준비되여있는데 바로 이 기판번호를 지정해주어야 한다.

영상방식값은 기판마다 미리 정해져있다.

례를 들어 VGA기판의 기판번호는 9, 방식의 값은 0부터 2까지의 값을 가진다. 그가운데서 2방식이 640×480으로서 분해능이 제일 높다. 이것은 화면에서 그림을 표시하는 점의 개수가 가로 640, 세로 480이라는것을 의미한다.

사용자들이 기판종류를 잘 모를 때는 0 또는 DETECT로 하면 C언어처리기가 자동적으로 기판번호를 선택해준다.

프로그람을 작성할 때 영상방식값을 지정하지 않으면 C언어처리기는 영상표시기판이 가지고있는 방식값가운데서 분해능이 제일 높은것을 자동적으로 선택한다.

도형그리기프로그람에서는 또한 영상표시기판을 리용하기 위하여 준비되여있는 영상표시기판구동파일을 지적해주어야 한다. 영상표시기판구동파일에는 cga.bgi, vga.bgi 등이 있다.

2. 자리표계

여러가지 도형을 화면에 그릴 때 그리려는 도형의 몇개의 자리표를 지정하여 그린다. 례를 들어 선분을 그리려면 선분의 두 끝점의 자리표를 지정하여주고 원을 그리려면 원의 중심과 반경을 지정하여준다.

그러므로 화면에 요구되는 도형을 그리자면 화면의 자리표계가 어떻게 이루어져 있는가, 그것을 어떻게 변화시킬수 있는가를 알아야 한다.

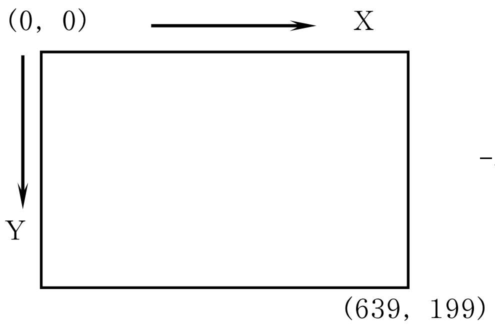
그라프화면자리표계

도형을 그리는 자리표계에는 화면자리표계와 가상자리표계가 있다.

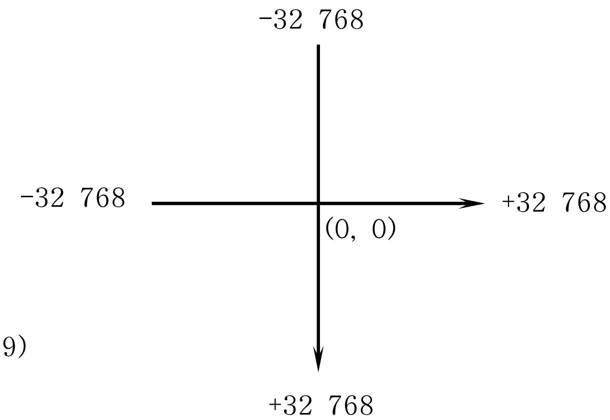
화면자리표계는 현시장치화면우에 설정하는 자리표계이고 가상자리표계는 콤퓨터내부에서 그림을 그릴 때 리용하게 되는 자리표계이다.

화면자리표계는 왼쪽웃구석점을 자리표원점 (0, 0)으로 한다.

x축은 왼쪽에서 오른쪽으로 향한 방향이 정방향이고 y축은 우에서부터 아래로 내려오는 방향이 정방향이다.



640×200방식인 경우의 화면자리표계

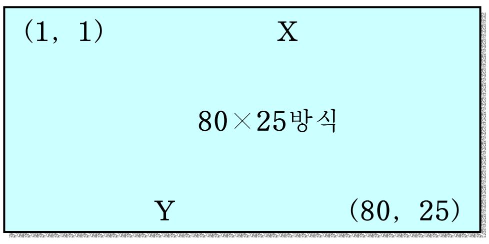


가상자리표계

C언어로 그림을 그릴 때 가상자리표한계범위안에서는 오유없이 그림이 그려지나 실지 우리 눈에는 화면자리표계범위안에 놓인 부분만이 보인다. 따라서 그림이 화면자리표계범위밖에 놓일 때에는 자리표를 고치거나 그림을 확대, 축소하는 방법으로 화면자리표계범위안에 놓이도록 해야 한다.

기호화면자리표계

기호화면에도 자리표계를 설정할수 있는데 자리표계의 최대한계값은 기호화면방식에 관계된다. 기호화면방식이란 한화면에 쓸수 있는 기호수와 기호의 색을 지정하는것을 말한다.



기호화면방식은 textmode(방식값)로 지정한다.
방식값에는 0 - 3, 7, 256이 있다.

기호화면방식값

방식값	화면방식	방식값	화면방식
0	40×25흑백	3	80×25색
1	40×25색	7	80×25단색
2	80×25흑백	256	40×49색

례를 들어 textmode(3)이라고 하면 한화면에 25행, 80렬의 기호를 색으로 쓸수 있다.

3. 색의 조종

모든 천연색영상방식들은 조색판을 가지고있다.

CGA는 미리 정의된 고정된 색갈모임을 가지는 4개의 서로 다른 조색판을 가지고 있다.

EGA, VGA, MCGA영상표시기판들은 사용자의 요구에 맞게 다시 정의할수 있는 조색판을 가지고있다.

설치된 영상표시기판과 현재의 영상방식에 따라서 화면상에서 2, 4, 8, 16, 256개의 색갈을 현시할수 있다.

색갈은 색첨수를 선택하는 방법으로 지정한다. 색첨수를 화소점값 혹은 색속성이라고 한다. 색첨수는 0부터 n-1까지의 옹근수값을 가질수 있다. 여기서 n은 조색판에서 색갈의 수이다.

C언어에서 사용하는 그리기색은 모두 16가지의 색인데 매 색에 해당한 색첨수값들을 다음의 표에 주었다.

색 첨 수

번호	색이름	번호	색이름	번호	색이름	번호	색이름
0	검은색	4	붉은색	8	검은 재색	12	밝은 붉은색
1	남색	5	분홍색	9	밝은 남색	13	밝은 분홍색
2	풀색	6	밤색	10	밝은 풀색	14	노란색
3	청색	7	밝은 재색	11	밝은 청색	15	흰색

제2절. 간단한 도형그리기

1. 점, 선, 기타 도형그리기

실례프로그람과 그에 대한 해설을 통하여 그림그리는 방법을 학습하기로 한다.

례 1: 화면중심에 붉은색동그라미를 그리는 프로그람

```
{1}     #include <graphics.h>
{2}     #include <stdio.h>
{3}     #include <conio.h>
{4}         int main(void)
{5}         {
{6}         int i;
{7}         int gd=DETECT;
{8}         int gm;
{9}         initgraph(&gd, &gm, " ");
{10}        setcolor(4);
{11}        circle(320, 240, 50);
{12}        getch();
{13}        closegraph();
{14}        return 0;
{15}        }
```

이 프로그람이 실행되면 화면중심에 붉은색동그라미가 그려진다.

해 설

{1}: graphics.h파일을 불러들인다.

그림을 그리자면 그림을 그리는데 필요한 모든 기능을 갖추고있는 graphics.h라는 파일이 필요하다.

{9}: 그림을 그릴수 있는 환경을 조성시키는 명령문이다. 화면에 그림을 그리자면 이 명령문(그라프환경설정명령문)을 반드시 리용해야 한다.

이 명령문에서 gd는 영상표시기판종류를 선택하기 위한것이다.

gm은 프로그람화면방식을 지정하기 위한것이다.

례를 들어 VGA기판에서 기판번호는 9, 방식의 값은 0부터 2까지의 값을 가진다. 그가운데서 2방식이 640×480으로서 분해능이 제일 높다. 이것은 화면에서 그림을 표시하는 점의 개수가 가로 640, 세로 480이라는것을 의미한다.

프로그람을 작성할 때 gm을 지정하지 않으면 언어처리기는 영상표시기판이 가지고있는 방식값가운데서 분해능이 제일 높은것을 자동적으로 선택한다.

" "는 영상표시기판구동기파일이 들어있는 구동기를 지적하는것이다.

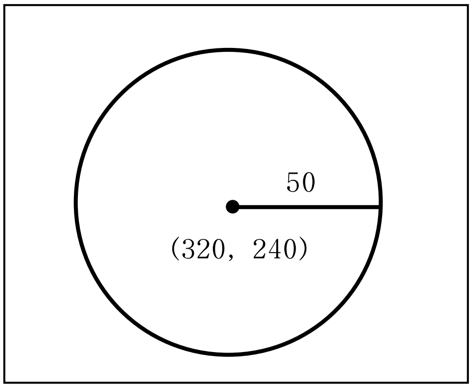
{10}: 동그라미의 색갈을 붉은색으로 지정하는 명령문이다.

{11}: 동그라미를 그리는 명령문이다.

이 행에서 320, 240은 원의 중심자리표이고 50은 반경이다.

{13}: 그라프환경에서 벗어나 수자나 문자만을 표시하는 기호화면방식으로 넘기는 명령문이다.

이 명령문이 실행되면 그라프환경이 닫겨지게 된다.



graphics.h에 선언되여있는 함수

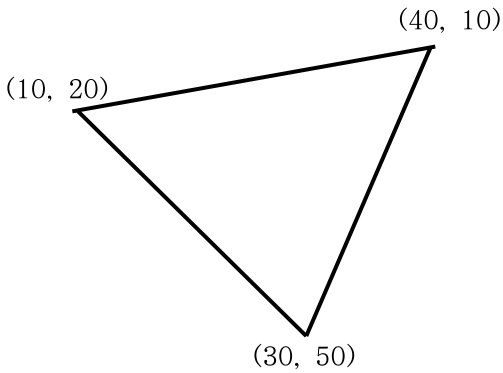
함 수 선 언	기 능
void arc(int x, int y, int stangle, int endangle, int radius);	활등을 그린다.
void bar(int left, int top, int right, int bottom);	2차원도형을 그린다.
void bar3d(int left, int top, int right, int bottom, int depth, int topflag);	3차원도형을 그린다.
void circle(int x, int y, int radius);	(x, y)를 중심으로 하는 원을 그린다.
void cleardevice(void);	도형화면을 지운다.
void closegraph(void);	도형그리기를 끝낸다.
void drawpoly(int numpoints, int *polypoints);	다각형을 그린다.
void ellipse(int x, int y, int stangle, int endangle, int xradius, int yradius);	타원을 그린다.
void fillellipse(int x, int y, intxradius, int yradius);	타원을 그리고 내부를 색칠한다.
void fillpoly(int numpoints, int *polypoints);	다각형을 그리고 내부를 색칠한다.
int getcolor(void);	현재 펜의 색갈을 얻는다.
void getimage(int left, int top, int right, int bottom, void *bitmap);	지정된 구역의 화상을 기억기에 보존한다.
int getmaxx(void);	화면의 제일 큰 x자리표값을 돌려준다.
int getmaxy(void);	화면의 제일 큰 y자리표값을 돌려준다.
unsigned imagesize(int left, int top, int right, int bottom);	화상을 보존하는데 필요한 byte수를 돌려준다.
void initgraph(int *graphdriver, int *graphmode, char *pathtodriver);	그림그리기를 초기화한다.
void line(int x0, int y0, int x1, int y1);	두 점사이에 직선을 긋는다.

함수선언	기능
void outtextxy(int x, int y, char *textstring);	지정된 위치에 문자렬을 출력한다.
void pieslice(int x, int y, int stangle, int endangle, int radius);	부채형을 그리고 내부를 색칠한다.
void putimage(int x, int y, void *bitmap, int top);	화면에 화상을 출력한다.
void rectangle(int left, int top, int right, int bottom);	직4각형을 그린다.
int registerbgidriver(void (*driver)(void));	이미 련결된 도형구동프로그람 코드에 들어간다.
void setcolor(int color);	현재 펜의 색갈을 설정한다.
void setfillstyle(int pattern, int color);	내부색칠하기방식과 색갈을 설정한다.
void setlinestyle(int linestyle, unsigned upattern);	펜의 굵기와 류형을 설정한다.
void settextjustify(int horiz, int vert);	도형함수를 위한 본문의 맞추기방식을 설정한다.
void settextstyle(int font, int direction, char size);	도형출력을 위해 현재의 본문속성을 설정한다.
void setwritemode(int mode);	도형방식에서 선그리기의 출력방식을 설정한다.

례 2: 화면우에 세 점 (30, 50), (10, 20), (40, 10)을 정점으로 하는 풀색 3각형을 그리는 프로그람

```
#include <graphics.h>
#include <stdio.h>
#include <conio.h>
int main(void)
{
int gd=DETECT;
int gm;
initgraph(&gd, &gm, " ");
setcolor(2);
line(30, 50, 10, 20);
line(10, 20, 40, 10);
line(40, 10, 30, 50);
getch();
closegraph();
```

```
return 0;
}
```



이 프로그람이 실행되면 콤퓨터화면에 풀색으로 3각형이 그려진다.

프로그람에서 10행의 line문은 두 점 (30, 50)과 (10, 20)사이의 직선을 긋는 명령문이다.

례 3: 화면중심에 분홍색으로 임의의 날자를 쓰는 프로그람

```
{1}   #include <graphics.h>
{2}   #include <stdio.h>
{3}   #include <conio.h>
{4}   int main(void)
{5}    {
{6}   char day[30]="2012/4/15";
{7}   int gd=DETECT;
{8}   int gm;
{9}   initgraph(&gd, &gm, " ");
{10}  textcolor(5);
{11}  cleardevice();
{12}  restorecrtmode();
{13}  gotoxy(30, 12);
{14}  cprintf("%s", day);
{15}  getch();
{16}  return 0;
{17}  closegraph();
{18}  }
```

해 설

이 프로그람이 실행되면 화면중심에 입력한 날자가 표시된다.

{3}은 기호화면에 대한 조작을 진행하는 기능들이 정의되여있는 conio.h파일을 리용할수 있게 한다. 기호화면은 문자나 수자만 표시할수 있는 화면이다.

{10}은 기호의 색을 분홍색으로 표시하도록 한다.
{12}는 그라프화면방식으로부터 기호화면방식에로 돌려보내는 명령문이다.
{13}은 기호화면자리표계의 점 (30, 12)에로 유표를 이동시키는 명령문이다.

례 4: 붉은색 오각별을 그리는 프로그람
두가지 방법으로 그려보자.

[프로그람1]

```
#include <graphics.h>
#include <stdio.h>
#include <conio.h>
int main(void)
{
    int gd=DETECT;
    int gm;
    initgraph(&gd, &gm, " ");
    setcolor(12);
    line(291, 88, 201, 364);
    line(291, 88, 392, 364);
    line(170, 176, 422, 176);
    line(201, 364, 422, 176);
    line(170, 176, 392, 364);
    setfillstyle(1, 12);
    floodfill(293, 157, 12);
    floodfill(217, 194, 12);
    floodfill(317, 234, 12);
    floodfill(251, 280, 12);
    floodfill(331, 280, 12);
    floodfill(350, 206, 12);
    getch();
    closegraph();
    return 0;
}
```

[프로그람2]

```
#include <graphics.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
```

```
int main(void)
{
   int a, i, x, y, x1, y1;
   double pi=3.14;
   int gd=0;
   int gm;
   initgraph(&gd, &gm, " ");
   setcolor(4);
   a=90;
   for(i=1; i<6; i++)
   {
      x=320+(cos(a*pi/180)*50);
      y=240-(sin(a*pi/180)*50);
      x1=320+(cos((a+144)*pi/180)*50);
      y1=240-(sin((a+144)*pi/180)*50);
      line(x, y, x1, y1);
      a=a+144;
   }
   setcolor(0);
   circle(320, 240, (50/3));
   setfillstyle(1, 4);
   floodfill(320, 240, 4);
   getch();
   closegraph();
   return 0;
}
```

례 5: 휘날리는 공화국기발을 그리는 프로그람

```
#include <graphics.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
int main(void)
{
   int a, i, x, y, x1, y1;
   double pi=3.14;
   int gd=DETECT;
   int gm;
```

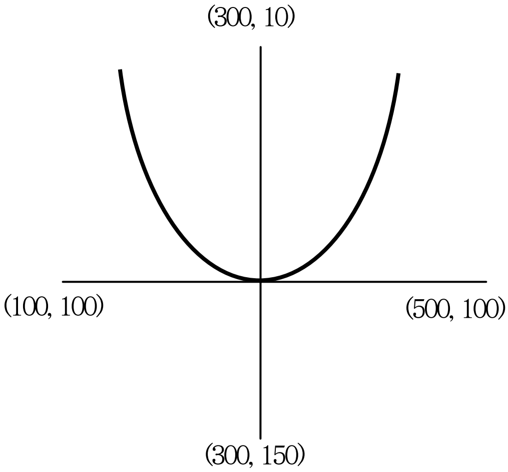
```
    initgraph(&gd, &gm, " ");
    setfillstyle(1, 1); bar(100, 100, 500, 300);
    setfillstyle(1, 15); bar(100, 140, 500, 260);
    setfillstyle(1, 4); bar(100, 150, 500, 250);
    setfillstyle(1, 15); fillellipse(240, 200, 40, 40);
    setcolor(4);
    a=90;
    for(i=1; i<6; i++)
    {
       x=240+(cos(a*pi/180)*40);
       y=200-(sin(a*pi/180)*40);
       x1=240+(cos((a+144)*pi/180)*40);
       y1=200-(sin((a+144)*pi/180)*40);
       line(x, y, x1, y1);
       a=a+144;
    }
    setcolor(0);
    circle(240, 200, (40/3));
    setfillstyle(1, 4); floodfill(240, 200, 4);
    getch();
    return 0;
    closegraph();
}
```

례 6: x가 -5에서 +5까지 0.5씩 변할 때 함수 $y=x^2$의 그라프를 그리는 프로그람

```
#include <graphics.h>
#include <stdio.h>
#include <conio.h>
int main(void)
 {
    double x, y;
    int gd=DETECT;
    int gm;
    initgraph(&gd, &gm, " ");
    line(100, 100, 500, 100);
    line(300, 10, 300, 150);
    x=-5;
    while(x<=5)
```

```
{14}        {
{15}            y=x*x;
{16}            putpixel(300+(x*20), 100-(y*10), 2);
{17}            x=x+0.5;
{18}        }
{19}        getch();
{20}        return 0;
{21}        closegraph;
{22}    }
```

이 프로그람이 실행되면 화면중심에 함수 y=x^2의 그라프가 나타난다.



해 설

{10}, {11}은 각각 가로축과 세로축을 그리는 명령문이다.

{16}은 화면자리표가 (300+(x*20), 100-(y*10))인 자리에 풀색의 점을 찍는다.

어떤 방정식에 따라 점들이 촘촘히 찍혀지면 마지막에는 그라프가 그려진다.

이 프로그람에서 {12} - {18}은 x를 -5에서 +5까지 0.5씩 증가시키면서 x^2함수의 곡선을 그리는 부분이다.

그리고 x자리표에 300을 더해주고 100에서 y자리표를 던것은 그라프를 화면중심에 그리기 위해서 자리표변환을 한것이다.

례 7: 각이 0° 부터 360° 까지 변할 때 sinx, cosx의 곡선을 그리는 프로그람

```
#include <graphics.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
int main(void)
```

```
{
   int i, js, jc, x, y, errorcode;
   double r, s, c, pi=3.14;
   int gd=DETECT;
   int gm;
   initgraph(&gd, &gm," ");
   errorcode=graphresult();
   if (errorcode!=grOk)          /* an error occurred */
   {
      printf("Graphics error:%s\n", grapherrormsg(errorcode));
      printf("Press any key to halt:");
      getch();
      exit(1);                   /* terminate with an error code */
   }
   /* x축표시 */
   line(0, 250, 400, 250);
   x=0;
   while(x<=360)
   {
      r=pi/180*x; i=(r*20);
      line(i+10, 247, i+10, 253); x=x+90;
   }
   /* y축표시 */
   line(10, 100, 10, 400);
   y=150; while(y<=350)
   {
      line(6, y, 14, y); y=y+25;
   }
   /* sinx, cosx그라프그리기 */
   for(x=0; x<=360; x++)
   {
      r=pi/180*x;
      s=sin(r); c=cos(r);
      i=(int)(r*20); js=(int)(s*50); jc=(int)(c*50);
      putpixel(i+10, 250-js, 4); putpixel(i+10, 250-jc, 2);
   }
   getch();
   return 0;
   closegraph();
}
```

이 프로그람이 실행되면 화면에 x, y축이 표시되고 그우에 붉은색 sinx그라프와 풀색의 cosx그라프가 그려진다.

2. 동화상처리

움직이는 그림을 그리는 프로그람수법에는 여러가지가 있다.

그가운데서 많이 쓰이는것이 동적기억기에 그림을 기억시켜놓고 그것을 화면우에서 이동경로를 따라 표시했다 지웠다 하는 조작을 반복하면 그림이 움직이는것처럼 보이게 하는 수법이다.

우리가 늘쌍 보는 영화도 바로 1초동안에 24장의 필림을 련이어 펼쳐보임으로써 마치도 움직이는것처럼 보인다.

그림의 기억과 꺼내기

imagesize(x1, y1, x2, y2): 점 (x1, y1)과 (x2, y2)를 대각점으로 하는 4각형안의 바이트수를 주는 함수

getimage(x1, y1, x2, y2, b): 점 (x1, y1)과 (x2, y2)를 대각점으로 하는 4각형안의 그림을 b로 지적한 이름으로 비트별로 기억시키는 함수

putimage(x, y, b, bb): getimage로 기억시켰던 직4각형안의 비트영상을 점 (x, y)를 왼쪽 웃구석점으로 하는 4각형안에 표시한다. bb는 비트영상을 화면에 표시하는 방식을 지정하는 상수이다.

```
Nomalput = 0   {mov}
Xorput = 1     {xor}
Orput = 2      {or}
Andput = 3     {and}
Notput = 4     {not}
```

bb=0으로 설정하면 b에 기억된 비트영상을 그대로 표시한다. 그외 다른 상수들은 b에 기억된 비트영상과 현재 화면에 그려진 비트영상을 비트연산(xor, or, and, not)하여 얻어진 결과로 표시한다.

실례프로그람과 그에 대한 해설을 통하여 움직이는 그림을 그리는 방법을 학습하기로 하자.

례 1: 단색공이 직선운동하는 모양을 그리는 프로그람

```
#include <graphics.h>
#include <stdio.h>
#include <conio.h>
#include <alloc.h>
#include <dos.h>
int main(void)
{
   int i, n, *p;
   int gd=DETECT;
```

```
        int gm;
        initgraph(&gd, &gm, " ");
        setcolor(10);
        circle(320, 240, 30);
        n=imagesize(290, 210, 350, 270);
        p=(int*)malloc(n*sizeof(int));
        getimage(290, 210, 350, 270, p);
        for(i=1; i<=30; i++)
        {
            putimage(i*20, 210, p, 0);
            clearviewport();
            delay(10000);
        }
        free(p);
        getch();
        closegraph();
        return 0;
    }
```

해 설

프로그람에서 15행은 프로그람을 기억할 동적기억구역을 확보하는 명령문이다.

여기서 p는 이 구역의 첫 주소를 기억하는 변수로서 이 변수는 지적자형변수이다. n은 확보할 기억바이트수인데 이 값은 그림의 크기에 따라 결정된다.

15행은 그림을 동적기억구역에 p라는 이름을 달아 기억시키는 명령문이고 19행은 기억시켰던 그림을 다시 화면에 표시한다.

21행은 화면에 표시된 공이 일정한 시간동안 나타나있도록 하기 위한 조작이다.

이 프로그람이 실행되면 화면에서는 밝은 풀색원이 왼쪽에서 오른쪽으로 옮겨가게 된다.

례 2: 붉은색 오각별이 화면의 왼쪽에서부터 오른쪽으로 서서히 움직이도록 하는 프로그람

```
#include <graphics.h>
#include <stdio.h>
#include <conio.h>
#include <alloc.h>
#include <dos.h>
int main(void)
```

```
{
    int x, y, i, *s, size;
    int gd=DETECT;
    int gm;
    initgraph(&gd, &gm, " ");
    setcolor(12);
    line(291, 88, 201, 364);
    line(291, 88, 392, 364);
    line(170, 176, 422, 176);
    line(201, 364, 422, 176);
    line(170, 176, 392, 364);
    setfillstyle(1, 12);
    floodfill(293, 157, 12);
    floodfill(217, 194, 12);
    floodfill(317, 234, 12);
    floodfill(251, 280, 12);
    floodfill(331, 280, 12);
    floodfill(350, 206, 12);
    size=imagesize(170, 88, 422, 364);
    s=(int*)malloc(size*sizeof(int));
    getimage(170, 88, 422, 364, s);
    x=50; y=150;
    do{
        clearviewport();
        putimage(x, y, s, 0);
        x=x+10;
        delay(10);
      }
    while(x<300);
    free(s);
    getch();
    closegraph();
    return 0;
}
```

해 설

이 프로그람에서 8행, 26행, 27행, 31행에서 밑줄그은 명령문은 오각별을 이동시키기 위한 조작이다. 앞의 실례에서와는 달리 오각별이 서서히 움직이도록 하기 위하여 delay문을 쓰지 않았다.

련습문제

1. 화면에 구명대와 같은 고리를 4개 그리고 서로 다른 색, 서로 다른 무늬로 내부를 색칠하여라.

2. 빛을 뿌리는 태양을 그리여라.

3. 반경이 70인 원을 그리고 그우에 중심점이 놓이면서 반경이 7인 원들을 그리여라.

4. 임의의 건을 누를 때까지 시간과 날자를 돌려주는 프로그람을 작성하여라.(100분의 1초까지)

5. x가 0부터 4π까지 변할 때 sinx, cosx곡선을 붉은색, 푸른색으로 그리여라.(x축과 y축도 표시하여라.)

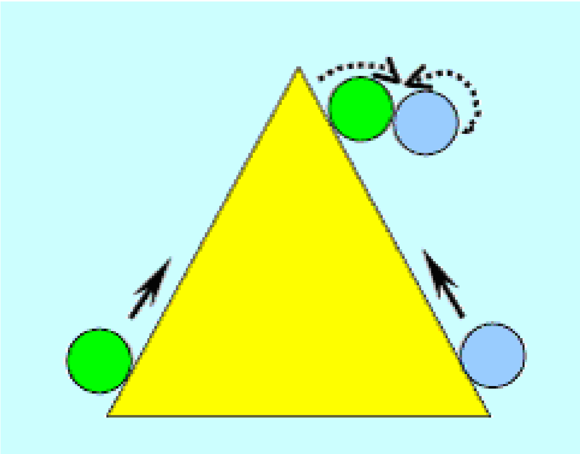
6. 화면에 서로 다른 색갈과 문양으로 된 직4각형탑을 7층까지 쌓아라.

7. 화면에 My Computer를 표시하고 그것이 서서히 커졌다 다시 작아지도록 하여라.

8. 각 x가 0° 부터 90° 사이에서 1° 간격으로 변할 때 tanx곡선을 그리여라.(x, y축우에 눈금척도를 매기여라.)

9. n을 입력시키고 nπ주기동안의 시누스곡선을 그리여라.

10. 두개의 공이 3각형의 세 변을 따라 서로 마주향하여 계속 돌아가다가 서로 만나면 그림의 점선과 같이 접하여 돈다. Enter건을 누르는 순간 정지하도록 하는 프로그람을 작성하여라.



11. 화면중심에 《경축》이라는 글발이 1초간격으로 현시되면서 동시에 축포가 터져오르는 명절의 밤을 형상하여라.

12. →, ←, ↑, ↓건을 누르는 동안 그에 비례하여 단색공이 왼쪽, 오른쪽, 우, 아래로 일정한 거리만큼 이동하도록 하는 프로그람을 작성하여라.

13. 초침, 분침, 시침이 있는 시계의 운동과정을 펼쳐보이는 프로그람을 작성하여라.

14. 《배움의 천리길》로정도를 그리는 프로그람을 작성하여라.

부 록

string.h에 선언되여있는 함수

함 수 선 언	기 능
char *strcpy(char *s, const char *t);	문자렬 t를 s에 복사하고 s를 돌려준다.
char * strncpy(char *s,const char *t, size_t n);	문자렬 t를 s에 n문자만큼 복사하고 s를 돌려준다.
char * strcat(char *s, const char *t);	문자렬 s의 뒤에 문자렬 t를 련결하고 s를 돌려준다.
char * strncat(char *s, const char *t, size_t n);	문자렬 s의 뒤에 문자렬 t를 최대 n문자 련결하고 s를 돌려준다.
int strcmp(const char *s, const char *t)	문자렬 s와 문자렬 t를 비교하고 문자렬 s < 문자렬 t 인 때는 부수 문자렬 s = 문자렬 t 인 때는 0 문자렬 s > 문자렬 t 인 때는 정수 를 돌려준다.
int strncmp(const char *s, const char *t , size_t n);	문자렬 s와 문자렬 t를 최대 n문자만큼 비교한다. 비교결과는 strcmp()와 같다.
char *strchr(const char *s, int c);	문자렬 s에서 처음에 나타나는 문자에로의 지적자를 돌려준다. 문자렬 s에 문자 c가 없으면 NULL을 돌려준다.
size_t strlen(const char *s);	문자렬 s의 길이를 돌려준다.

ctype.h에 선언되여있는 함수

함 수 선 언	기 능
int isdigit(int c);	문자 c가 10진수이면 참을 돌려준다.
int islower(int c);	문자 c가 소문자이면 참을 돌려준다.
int isupper(int c);	문자 c가 대문자이면 참을 돌려준다.
int isalpha(int c);	문자 c가 영문자이면 참을 돌려준다.
int isalnum(int c);	문자 c가 영문자, 수자이면 참을 돌려준다.
int isprint(int c);	문자 c가 표시가능한 문자이면 참을 돌려준다.
int tolower(int c);	문자 c가 대문자이면 소문자로 변환한다.
int toupper(int c);	문자 c가 소문자이면 대문자로 변환한다.

math.h에 선언되여있는 함수

함 수 선 언	기 능
double sin(double x)	sin x
double cos(double x)	cos x
double tan(double x)	tan x
double asin(double x)	$\sin^{-1} x$
double acos(double x)	$\cos^{-1} x$
double atan(double x)	$\tan^{-1} x$
double sinh(double x)	sinh x
double cosh(double x)	cosh x
double tanh(double x)	tanh x
double exp(double x)	e^x
double log(double x)	$\log_e x$
double log10(double x)	$\log_{10} x$
double pow(double x, double y)	x^y
double sqrt(double x)	$\sqrt{x}$
double fabs(double x)	$\lvert x \rvert$

stdlib.h에 선언되여있는 함수

함 수 선 언	기 능
int atoi(const char *s);	문자렬로 표시된 값을 int형으로 변환
long atol(const char *s);	문자렬로 표시된 값을 long형으로 변환
double atof(const char *s);	문자렬로 표시된 값을 double형으로 변환
int abs(int x);	$\lvert x \rvert$
long labs(long x)	$\lvert x \rvert$ (long형)
int rand(void);	0 ~ 32 767범위의 우연수를 발생시킨다.
void exit(int state);	프로그람을 정상완료시킨다.
void abort(void);	프로그람을 이상완료시킨다.

전 처 리

C언어원천프로그람은 두개의 부분으로 이루어졌다. 한 부분은 번역되기 전에 처리해야 할 명령들과 다른 한 부분은 변수정의, 함수선언, 함수호출 등과 같이 번역될 때와 실행시에 처리해야 할 명령들이다.

원천프로그람이 정식으로 번역되기 전에 처리해야 할 명령을 전처리명령 혹은 간단히 전처리라고 하며 이것은 전처리기(preprocessor)에 의하여 수행된다.

전처리를 거쳐 만들어진 새로운 원천프로그람은 번역과정을 거쳐 비로소 자기의 목적을 달성하게 된다.

C언어의 전처리에는 다음과 같은것들이 있다.

- 마크로정의
  - #define
  - #undef
- 파일포함
  - #include
- 조건번역
  - #if
  - #ifdef
  - #else
  - #elif
  - #endif
- 기타
  - #line
  - #error
  - #pragma

- 마크로정의에는 두가지 즉 파라메터가 없는 마크로정의와 파라메터가 있는 마크로정의가 있다.

파라메터 없는 마크로정의형식: #define macro_name macro_content

파라메터있는 마크로정의형식: #define macro_name(파라메터렬) macro_content

여기서 macro_name은 마크로이름으로서 반드시 합법적인 표식부호여야 하며 일반적으로 대문자로 명명한다. macro_content는 마크로의 본문내용이다.

macro_name과 macro_content사이는 반드시 공백으로 분리한다.

마크로는 프로그람작성자가 고의적으로 하지 않는 경우에는 일반적으로 반두점으로 끝을 맺지 않는다. 전처리에서는 다만 본문의 치환조작만 하고 마크로의 내용본문에 대해서는 처리를 진행하지 않기때문에 프로그람작성자가 마크로를 정의할 때 만일

마크로가 식이라면 반드시 이 식을 ()로 묶어주어야 한다. 그렇지 않으면 예측할수 없는 결과가 발생한다. 파라메터있는 마크로를 사용할 때 파라메터의 대부분이 식이고 마크로내용도 식이므로 전체 마크로내용을 묶어주어야 할뿐아니라 마크로파라메터도 ()로 묶어주어야 한다.

마크로정의에서 지켜야 할 점

① 마크로정의명령문은 일반적으로 함수밖에 놓여야 하며 반드시 단독적으로 한행을 차지해야 한다.

② 프로그람작성자가 마크로를 정의할 때 반드시 #define명령의 마지막에 Enter건을 눌러야 한다. 그렇지 않으면 번역오유를 발생시킬수 있다.

③ 마크로의 작용범위는 그 마크로정의의 #define명령으로부터 시작해서 파일의 끝이나 #undef명령을 만나 마크로를 취소할 때까지이다.

전처리명령 #undef의 작용은 하나의 마크로정의를 취소하는것이다. 그의 형식은 다음과 같다.

```
#undef macro_name
```

- 파일포함은 C원천코드가 #include명령을 통해서 다른 파일(보통 .c혹은 .h파일)의 전체 내용을 포함하도록 하는것이다.

형식: #include "파일이름" 혹은 #include <파일이름>

전처리기의 파일포함처리단계

① 포함되는 파일의 본문내용을 원천코드의 #include명령위치에 삽입하여 새로운 원천코드를 형성한다.

② 만일 포함되는 파일에도 #include명령이 있으면 먼저 그 #include명령을 처리한다.

만일 #include명령에서 파일이름을 < >으로 묶었으면 전처리기는 C체계의 파일포함목록에서 포함되는 파일을 찾는다.

만일 #include명령문의 파일이름을 " "로 묶었으면 번역기는 먼저 C체계를 기동한 등록부에서 포함되는 파일을 찾고 만일 없으면 체계의 파일포함목록에서 포함되는 파일을 찾는다.

- 조건번역명령에는 대체로 3가지형식이 있다.

많은 경우 이식성을 높이기 위해 C언어원천코드에는 각종 코드토막이 포함된다. 그러나 어떤 경우에는 한 토막의 코드에 대해서만 번역을 진행해야 할것을 요구한다. 이때에 조건번역명령을 리용해야 한다.

① **#if ~ #endif정의형식**:

```
#if조건 1
        원천코드토막 1
#elif조건 2
        원천코드토막 2
#else
```

```
        원천코드토막 3
#endif
```

② #ifdef ~ #endif정의형식:

```
#ifdef 마크로이름 1
        원천코드토막 1
#elif defined 마크로이름 2
        원천코드토막 2
#else
        원천코드토막 3
#endif
```

③ #ifndef ~ #endif정의형식:

```
#ifndef 마크로이름
        원천코드토막 1
#else
        원천코드토막 2
#endif
```

여기서 elif는 else if를 줄여서 쓴것이다.

#elif와 #else는 없어도 되지만 #endif는 반드시 있어야 한다. 그것은 #if명령의 끝이다. #elif명령은 여러개 있을수 있다. #if뒤에는 괄호를 치지 않는다.

어떤 조건이 만족될 때 이 조건아래의 원천코드토막은 유효하며 그렇지 않으면 무효로 된다. 조건은 반드시 상수식이여야 하는데 대체로 마크로이름을 쓴다.

매 명령은 독자적으로 한 행을 차지한다.

#if와 #elif는 보통 defined명령과 배합하여 사용한다. defined명령의 형식은

defined(마크로이름)　혹은　defined 마크로이름

defined명령의 기능은 어떤 마크로가 정의되였는가를 판단하는것이다. 만일 이미 정의되였다면 defined명령은 1을 돌려주고 그렇지 않으면 0을 돌려준다. defined명령은 #if 혹은 #elif와만 배합하여 사용할수 있으며 단독적으로 사용할수 없다. 실례로 #if defined(ABC)의 의미는 《만일 마크로 ABC가 정의되였다면》이다.





콤 퓨 터 상 식

프로그람개발에서 교환할수 없는 인원수와 시간

만약 어떤 프로그람을 개발하는데 6명의 개발자가 12달동안 걸렸다고 할 때 즉 6(명)×12(달)=72(인월)로 완성되였다고 하면 72(명)×1(달)=72(인월)라는데로부터 이것을 72명의 개발자를 동원하여 1달동안에 완성할수 있는가?

물론 이것은 불가능하다. 따라서 프로그람개발에서 작업량을 가늠할 때 《인월》만으로 추정하는것은 그리 의의가 없다.

프로그람개발일정의 지연을 만회하려고 몇십명의 인원을 더 투하하는 식으로 문제를 해결할수는 없는것이다.

이런 경우 그들에 대한 교육과 훈련에 투자가 더 들며 인원이 많아질수록 정보교환의 중복이 증가하게 되므로 오히려 개발기일이 지체될 가능성이 커진다.

이것을 《불룩스의 법칙》이라고 부르고있다.

첫째도 둘째도 유능한 인재가 기본이다.

프로그람개발의 생산성을 1명이 1달동안에 작성하는 프로그람의 행수로 측정한다면 《최고급》프로그람작성수와 《최하급》프로그람작성수사이에는 25배나 차이가 있다고 본다. 또한 품질의 견지에서도 프로그람 1 000행당의 오유발생률을 비교하면 최고급프로그람작성수와 최하급프로그람작성수사이에는 10배정도의 차이가 있다고 본다.

콤퓨터(제1중학교 제6학년용)

집 필	김병철, 김강호, 리은경, 리영걸	심 사	심의위원회
편 집	김해경		
장 정	김순영	교 정	김옥화
낸 곳	교육도서출판사	인쇄소	평양고등교육도서인쇄공장
인 쇄	주체99(2010)년 12월 1일	발 행	주체99(2010)년 12월 11일

교-10-보-801 값 10 원